To appear in the *Proceedings of the 1st Understanding Problems Workshop* at the 2012 Genetic and Evolutionary Computation Conference (GECCO-2012).

Assessment of Problem Modality by Differential Performance of Lexicase Selection in Genetic Programming: A Preliminary Report

Lee Spector
Cognitive Science
Hampshire College
Amherst, MA 01002
lspector@hampshire.edu

ABSTRACT

Many potential target problems for genetic programming are modal in the sense that qualitatively different modes of response are required for inputs from different regions of the problem's domain. This paper presents a new approach to solving modal problems with genetic programming, using a simple and novel parent selection method called lexicase selection. It then shows how the differential performance of genetic programming with and without lexicase selection can be used to provide a measure of problem modality, and it argues that defining such a measure in this way is not as methodologically problematic as it may initially appear. The modality measure is illustrated through the analysis of genetic programming runs on a simple modal symbolic regression problem. This is a preliminary report that is intended in part to stimulate discussion on the significance of modal problems, methods for solving them, and methods for measuring the modality of problems. Although the core concepts in this paper are presented in the context of genetic programming, they are also relevant to applications of other forms of evolutionary computation to modal problems.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program synthesis*; F.2.m [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Miscellaneous*

General Terms

Algorithms

Keywords

Genetic programming, modal problems, selection, lexicase selection, problem metrics, modality

1. INTRODUCTION

Part of the appeal of genetic programming (GP) [5] is its potentially broad applicability. Indeed the claim has often been made that GP can be applied to almost any problem for which the solution is a computer program. John Koza, in his 1992 book, refers to the problem of "discovering a computer program that produces some desired output when presented with particular inputs" as "the problem of program induction." He claims that this problem is pervasive across many fields, and he claims that GP is a general method for solving the problem of program induction [5, Chapter 2].

The class of problems for which the solution is a computer program is indeed very broad, and many problems that do not initially appear to meet this requirement can be recast in straightforward ways so that they do. But while one can readily see how GP can be applied to all such problems in principle, the problems to which it has been *successfully* applied have been more limited. More specifically, while successful applications of GP span many disciplines of science and engineering, and even reach into several areas in the arts and humanities, they are generally limited to problems for which solution programs can perform similar actions for all possible inputs. That is, there is a weaker track record on problems that require qualitatively different actions for different inputs; we will call such problems "modal" and use the term "modes" (or "modes of response") for the qualitatively different actions that a successful program must perform.[1]

Of course GP has long been applied to problems that allow for conditional execution, and in some cases one might consider the evolved programs to perform qualitatively different actions depending on inputs. For example, the standard 11-multiplexer problem [5, pp. 170–187] allows evolved programs to include an `IF` function and a solution can use this to "do something different" for some inputs than it does for others. An evolved solution might, for example, begin with an `IF` that tests address input bit `A2` and executes code for "selecting a low-order data bit" for one value or code for "selecting a high-order data bit" for the other. One could consider this problem to require two modes, a low-order mode and a high-order mode, and one could consider the evolved solution to be using a conditional operator to choose the appropriate mode of response.

This example does not, however, exemplify the kind of

[1]Note that "modal" as used here is quite different from "multimodal" as used elsewhere in the literature; we discuss this issue further in Section 2.

modal problem that is the goal of the present paper to address. It is not a trivial matter to say exactly why—and we discuss this difficulty in greater detail below—but perhaps an example of a more clearly modal problem will help the reader to see what is at issue.

Consider instead the problem of evolving a program that calculates a range of geometric formulae, including perimeters, areas, surface areas and volumes of a variety of two-dimensional and three-dimensional geometric shapes. One input would specify *which* formula the user wishes to calculate, while other inputs would specify numbers required for the specified calculation (for example lengths of edges). Some formulae may share components with one another—for example, formulae for circles and cones and spheres will share factors of π—but for the most part a solution program will have to do something qualitatively different depending on the `formula` input.

In many problems it will not be obvious what modes of response a problem requires, or how to tell when one or another is appropriate. For example, we can imagine a problem like the geometric formula problem in which there is no `formula` input, and in which the proper formula must be derived in some way from the values of other inputs. The concept of a "mode" implies that each situation (input combination) will require the system to respond in one mode or another (or in some combination of overlapping modes), but a particular mode of response will probably be correct for many different inputs. We may have no way of knowing, in advance, how many qualitatively different modes of response will be necessary or how to tell which particular situation calls for which particular actions from the system's repertoire. To make matters worse, modes may overlap in various ways. For example, consider a geometric formula problem that includes a hexadecimal mode that can be used in conjunction with the modes that we have already discussed.

Problems with this kind of modality may be particularly difficult for standard GP systems to solve because performance in one mode or combination of modes may be uncorrelated with performance in others. When one uses the most standard GP techniques (see below for exceptions) individuals are selected for reproduction and variation based on a simple aggregation of performance measures across all inputs. This may cause populations to become dominated by individuals that do well in some circumstances but poorly in others. If, as we are supposing, successful programs must "do something qualitatively different" in different circumstances, then standard GP techniques will not guide such populations toward complete solutions.

The class of problems with the kind of modality described here is vast, and arguably both much larger and much more significant than the class of problems on which GP has traditionally succeeded. One type of problem in which modality is ubiquitous, for example, is the programming of end-user software applications. Such applications usually perform multiple functions on a range of inputs of multiple types, and while the functions that they perform may share computational requirements they are also likely to be somewhat independent. Consider a web browser: it must display web pages but also create bookmarks, display browsing histories, negotiate secure connections, launch helper applications, and complete many other kinds of tasks. Many of these functions share the need to display HTML text, to manage user clicks, and to perform other common sub-tasks, but performance on the page display function is probably largely independent of performance on the bookmark creation function. Software engineers developing a browser are likely to use programming processes that are driven at least in part by suites of tests that measure performance function by function, and such tests are a reasonable starting point for the development of a fitness function that would allow GP to compete with human programmers in this area.[2] The satisfaction of such tests, however, is a highly modal problem in the sense discussed above; it is likely that the program will have to do qualitatively different things to satisfy different subsets of the tests. If we expect GP to perform well in such cases then it is essential that we first tackle the problem of evolving programs for modal problems.

In this paper we present a simple new approach for applying GP to modal problems. The approach involves a new parent selection method, called "lexicase selection," that is based on consideration of the performance of potential parents on individual fitness cases (input/output pairs). In short, we select a parent by filtering the population on the basis of performance on one fitness case at a time, with the fitness cases considered in random order (see below for details). We argue that GP with lexicase selection may be better able to solve modal problems than more traditional methods, and also that the improvement in problem solving performance that lexicase selection provides might be used as a measure of a problem's modality. The more general case that we want to make is that we need to continue to work on enabling GP systems to solve modal problems, and that quantitative measures of problem modality will be important tools in moving this work forward.

In the next section we give a few pointers into the large body of related work. This is followed by a section in which we describe lexicase selection in detail, and then a section in which we discuss ways in which lexicase selection performance can be used to provide a measure of modality. We then describe an illustrative example problem and present results with and without lexicase selection to show how this indicates the level of problem modality. This is followed by a discussion of whether it really makes sense to define a metric in terms of the success of a method that is intended to solve problems that score high on the metric. This will strike many readers as peculiar (or worse), but we will argue that it makes sense in the present context. We then conclude with a discussion of suggested future work.

2. RELATED WORK

Much of the prior related work has been conducted in the context of multi-objective evolution, and some of the techniques developed in this context may be applicable to modal problems if one considers performance in each mode to be a separate objective. However, in multi-objective problems one generally knows what the objectives are in advance, which may or may not be true for modal problems. In addition, we consider "modes" to be connected to circumstances (inputs) in a way that objectives need not be in general. For example, one might use multi-objective GP techniques to find a program that controls a truck backing up to a loading dock, simultaneously maximizing the objectives of speed,

[2]Presumably one would use only tests of top-level functions in a fitness test, since one would like the GP system, not a human software engineer, to perform the factoring that must precede specification of unit-tests for internal sub-functions.

safety, and fuel efficiency, but in this case all objectives apply to every test to which the program will be subjected.[3]

Nonetheless, many techniques developed for use on multi-objective problems may also be useful for modal problems. There is an enormous literature on solving multi-objective problems with evolutionary methods more generally, but a good summary of the state of the art in GP, as of 2008, can be found in [10, Chapter 9]. Most (but not all) of these techniques are focused, as is the proposal presented here, on modification of the parent selection algorithms used in the GP system. Some combine performance on all objectives into a single scalar metric, but this must often be done in a domain-specific way. Many others (e.g. [4]) involve consideration of the "Pareto front" of a population, which is the subset of individuals not "dominated" by any other individuals relative to the given objectives. An individual dominates another individual if it is at least as good as the other in all objectives and better than the other in at least one. Some prior work has combined selection based on membership in (or distance to) the Pareto front with other selection criteria (for example [6, 9]).

Some prior work on multi-objective GP altered the selection methods to consider objectives sequentially, or "lexicographically," with those considered earlier having priority and those considered later serving only to break ties; for example Luke and Panait did this with the objectives of correctness and program size, prioritizing correctness [8]. The lexicase technique described below incorporates the lexicographic concept but applies it to fitness cases and in randomized order.

The term "multimodal" has occasionally been used to mean something similar to what is meant here by "modal," and work has been conducted on evolving solutions to problems that are multimodal in this sense. None of this work, however, has provided an approach that is as general or compatible with standard GP techniques as the approach offered here. For example, the technique described in [1] applies only in the context of a novel architecture for evolving finite state automata. Much more commonly, the term "multimodal" has been used to describe problems with multiple global optima (e.g. [15]); this feature of problems is independent of "modality" as the term is used here.

To some extent the modes of response required by a problem can be considered to be separate subproblems. "Ensemble" methods [7, 2, 11, 13] can be used to solve problems through the aggregation of independent solutions to subproblems, but the present paper is concerned with the search for a single program that solves an entire modal problem. Note that in many applications, such as the geometry formula problem and the end-user applications discussed above, the sub-problems will share computational demands and therefore a solution consisting of a single modular program will often make more sense than an ensemble of programs for individual modes.

The mathematical notion of a "piecewise" definition is also related to modality insofar as modal problems will call for solutions that are in some sense defined in a piecewise fashion. But the mathematical literature does not provide us with a sufficiently general way to know when a piecewise solution is required, and it does not provide us with general methods for deriving piecewise solutions.

[3]A simpler version of this problem is described in [5].

3. LEXICASE SELECTION

The goal in developing lexicase selection was to provide a way to solve modal problems that is simple, tractable, problem-independent, and representation-independent while interfering as little as possible with other components of a GP system. The guiding assumption is that a problem's modality is likely to be factorable to some degree by its fitness cases—the input/output pairs that are used in assessing fitness—each of which presumably represents a "circumstance" with which a correct program must deal. Different (possibly unknown) subsets of the fitness cases may call for different modes of response; that is, they may require the system to respond in qualitatively different ways. This assumption holds for a variety of interesting modal problems, including the evolution of user-level applications based on tests of top-level functions which was discussed above.

Because we assume that a problem's modality is at least partly factorable by fitness case, we want to ensure that good performance with respect to any fitness case can be rewarded by the evolutionary process—presumably by allowing the high-performing individual to contribute to the following generation—regardless of performance on other cases. At the same time, however, it is important to reward progress on larger groups of cases and eventually on all of the cases; we seek a program that solves the entire modal problem, not independent programs for each of many problem components.

Lexicase selection meets these goals by substituting an alternative method for selecting parents into an otherwise standard (or arbitrarily non-standard) GP system. All other components of the system remain unchanged.

The core idea of lexicase selection is that a parent is selected by starting with a pool of potential parents and then filtering the pool on the basis of performance on individual fitness cases, considered one at a time. This sequential consideration of cases is reminiscent of the "lexicographic ordering" of character strings, in which the first character has the largest effect, the second matters only for ordering within the set of strings that are equal in their first characters, and so on. This similarity with lexicographic ordering, applied to the consideration of fitness cases, inspired the name "lexicase."

Different versions of lexicase selection can be obtained by varying the way in which the initial pool of potential parents is constituted, the way in which the sequence of cases is determined, and the way in which performance on a case is used to filter the pool. One of the simplest versions of lexicase selection, which might be more fully described as "global pool, uniform random sequence, elitist lexicase selection," is descibed in pseudocode in Figure 1.

The version of lexicase selection shown in Figure 1, which is also the version used in the experiments presented below, begins the selection of each parent with a pool consisting of the entire population (as shown in step 1a); this justifies its description as a "global pool" form of the algorithm. Alternatives include beginning with a subset of the population that is selected randomly or according to some spatial or "geographic" structure imposed on the population [14]. As shown in step 1b, this version of the algorithm re-orders the cases randomly each time it is called, selecting the orderings from a uniform distribution of all possible orderings; this justifies the description as using a "uniform random sequence." Alternatives might order the cases in a biased way, with

```
To select a parent for use in a genetic operation:

  1. Initialize:
       (a) Set candidates to be *the entire population.*
       (b) Set cases to be a list of all of the fitness cases *in random order.*
  2. Loop:
       (a) Set candidates to be the subset of the current candidates that have
           *exactly the best fitness* of any individual currently in candidates
           for the first case in cases.
       (b) If candidates or cases contains just a single element then return
           the first individual in candidates.
       (c) Otherwise remove the first case from cases and go to Loop.
```

Figure 1: Pseudocode for a simple version of the lexicase parent selection algorithm. A more complete description of this version of the algorithm is "global pool, uniform random sequence, elitist lexicase parent selection." Parts of the algorithm that can be varied to produce different versions of lexicase selection are indicated *in italics*; **see text for details.**

cases deemed to be more important or more difficult[4] being more likely to occur early in the list and thereby to have a greater influence on selection. Finally, as shown in step 2b this version of the algorithm allows only those individuals with the best fitness in the pool for the current fitness case to survive for further consideration, justifying the description of this version of the algorithm as "elitist." Alternatives here would be to allow individuals with fitnesses for the current case that are within some ϵ of the best in the group to survive, or to allow some pre-specified fraction of the group to survive.

The version of lexicase selection presented here samples the subset of the population's Pareto front that has the best fitness value in the population with respect to at least one fitness case. But it does not do so with a uniform distribution across such "Pareto elite" individuals. For example, if a large number of Pareto elite individuals have identical fitness for all cases then they will divide the chances for selection in selection events for which their elite cases occur first in the lexicographic ordering. Such sharing of chances for selection among individuals with identical fitnesses could be accomplished more simply by creating a pool of Pareto elite individuals, grouping individuals with identical fitnesses across all cases, choosing a group with a uniform distribution, and then choosing an individual from within the group with a uniform distribution. But lexicase selection also changes the selection distribution in other ways, among individuals with non-identical fitness values, in order to promote good performance on each fitness case and on each combination of fitness cases.

[4]The difficulty of cases might be determined dynamically over the course of a run, using a measure of "historically-assessed hardness" [3].

Table 1: Fitness components (errors by case, with smaller being better) and lexicase selection probabilities for a small population on a hypothetical problem.

	Fitness case				Lexicase selection
Individual	*a*	*b*	*c*	*d*	probability
#1	2	2	4	2	0.250
#2	1	2	4	3	0.000
#3	2	2	3	4	0.333
#4	0	2	5	5	0.208
#5	0	3	5	2	0.208

Consider a population containing only 5 individuals for a problem with 4 fitness cases, with the fitness components (which we will assume to be expressed as errors, for which lower is better) listed in Table 1.

In this population no two individuals are identical but every individual is Pareto elite; that is, no individual is dominated by any other individual and each has at least one fitness component that is the best in the population (although it may also be shared by others). For example individual #1 has the best fitness in the population for case *b* (shared with three other individuals), and it also has the best fitness in the population for case *d*. Not only do the lexicase selection probabilities vary across these individuals, but one individual (#2) has no chance of being selected at all. Examination of the probabilities in detail shows that they make a good deal of sense if we want to promote solutions to each case individually and also to groups of cases together.

Consider individual #2. It is elite for one case (case *b*), but it shares that elite status with three other individuals. Examining those three individuals we see that all of them are also elite for other cases, whereas individual #2 is not. So with all of the others we get elitism for case *b* combined with elitism in another case, and this makes all of the others more valuable than individual #2.[5]

Each of the other individuals is elite for exactly two cases, and yet they still vary in lexicase selection probability. To see why they do, and to see why it makes sense that they do, we have to look at the specific cases in which they are elite and of the performance of the rest of the population on all of the cases. For example, one reason for the high selection probability of individual #3 is that it is the *only* individual that is elite for case *c*. The lexicase selection algorithm selects individual #3 more often than the others because whenever case *c* is reached in a sequence of cases individual #3 will be the only remaining candidate. And it makes sense for individual #3 to have a higher probability of selection because we want to solve each individual case and individual #3 is the individual performing best on that case.

The differences in selection probability among individuals #1, #4, and #5 can be seen to arise from another factor that again makes sense relative to our goals of solving each individual case while also solving all combinations of cases. Notice that each of these individuals shares elite status with each of the others, but that individual #1 is better than either of the others on the one case for which none of them

[5]Other work in which selection is dependent on the number of cases for which individuals are elite is described in [12].

are elite. This explains why individual #1 is selected more frequently by the lexicase selection algorithm and it also accords with our goal of promoting individuals that are closer to solving all cases.

Notice also that the fact that the lexicase selection algorithm is sensitive to this difference between individual #1 and individuals #4 and #5 demonstrates that it considers all fitness component values (as it should), even though it may initially appear to be driven entirely by elite values. Non-elite values become important when the individuals having elite values for one case are eliminated early in a case sequence because they have inferior values on other cases.

What about the computational cost of lexicase selection? It is certainly more expensive than ordinary tournament selection, although its cost is well bounded for any particular population size and number of fitness cases. On the runs described below, the additional cost was not prohibitive and it was clearly outweighed by the benefits to guidance of the evolutionary search. On the other hand, with population size $10,000$ the overall cost of producing offspring using lexicase selection can exceed that of using ordinary tournaments by between one and two orders of magnitude. Various optimizations are possible, however, some of which might involve caching and some of which might involve performing roulette wheel selection with precomputed selection probabilities. It should also be noted that less global versions of lexicase selection will be less computationally expensive.

4. MEASURING MODALITY

The measurement of problem modality is tricky because it is not obvious how best to articulate or operationalize the most relevant notion of a "mode." As discussed above, the definition of modality should reflect the requirement that solutions must perform qualitatively different actions in different circumstances; that is, when they are presented with inputs (fitness cases) falling in different classes.

It would be tempting to consider the clustering of fitness cases by the similarity of inputs and/or outputs, and to consider the resulting clusters to reflect the required modes of response. But this may not capture the correct sense of "mode" because even when there are many such clusters, and even when they are quite distinct, it may nonetheless be the case that a single, possibly even unconditional function will map the inputs to the proper outputs across all fitness cases. Consider, for example, a symbolic regression problem in which the solution is a single polynomial but for which the fitness cases are clustered in several distinct sections of the input domains. Although clustering may make the problem appear to be modal, a successful program can perform similar (actually identical) actions for all possible inputs.

The kind of modality that we care about here is roughly the property of a problem that requires a solution to perform different actions for different inputs. But can this be measured directly? Could it perhaps be measured from the properties of solutions, after the fact? This too is tempting but notice that we could also produce, for *any* solvable problem, *some* solution that performs different actions for many sets of inputs or even for *each* different input. For example, we could produce a solution to a polynomial regression problem that uses different but semantically equivalent expressions for different input ranges, or even (with a sufficiently rich function set) fit different parts of the data with semantically distinct expressions. That is, we could produce a solution that incorporates more (but not less) modality than the problem demands. The modality of solutions is therefore an imperfect guide to the modality of problems.

It is not clear how to solve this problem completely, and we may have to settle for heuristic approximations. Indeed the suggestion presented here is for a heuristic approximation that, while imperfect, may provide useful information about the modality of a problem.

In particular, the suggestion here is based on the observation that the lexicase selection algorithm can reward good performance on any fitness case, and on any collection of fitness cases, regardless of performance on other cases or other sets of cases. It will not always reward good performance on every subset, but good performance on single cases and on small sets of cases will almost always be rewarded, and good performance on larger sets will also be rewarded with a frequency that is inversely related to the size of the set. Assuming that a problem's modality can be factored at least to some degree by fitness case—that is, that certain (possibly unknown) subsets of the fitness cases reflect performance in certain modes—this allows for programs to be rewarded for performing well in independent or partially independent modes. It also allows for programs to be rewarded for performing well in multiple modes, which may occur consecutively near the beginning of the list of cases in step 1b of the algorithm in Figure 1. In other words, if a problem is highly modal then lexicase selection should help a GP system to solve it. If this is true then the performance of a GP system that uses lexicase selection could be used as a measure of problem modality.

Of course, the "performance" of a GP system can itself be measured in many ways, including the success percentage (the percentage of runs that find solutions), the "computational effort" required to have a high probability of finding a solution [5], and the mean best fitness achieved over a collection of runs. Furthermore, what we really want to quantify is the difference between the performance of a system that uses lexicase selection and the performance of a system that doesn't, and there would also be many ways to calculate such a difference. For the sake of simplicity we will use the following measure of the modality (M) of a problem (p) relative to a GP system (λ):

$$M(p, \lambda) = \frac{S(p, \lambda_\alpha) - S(p, \lambda)}{S(p, \lambda_\alpha) + S(p, \lambda)} \qquad (1)$$

where S is the success rate for a given system on a given problem and λ_α is the system λ modified to use lexicase selection.

$M(p, \lambda)$ can range from -1 to 1. Values of less than 0 mean that lexicase selection makes the GP system perform *worse* than it would otherwise perform on the given problem, which might then be considered to be "anti-modal." While this might at first appear to be a rather strange concept, one kind of circumstance in which it would be reasonable to expect anti-modality is when *consistency* of responses is more important than the appropriateness of a particular response to a particular input; this might occur, for example, in a problem involving teamwork, in which consistent behavior allows teammates to predict and compensate for weak moves. Problems for which only a single mode of response is needed, but for which consistency is not more valuable than average good performance, should have values of $M(p, \lambda)$

Table 2: Successful runs (out of 30 in each condition) for standard GP and GP with lexicase selection on problems with three levels of modality. See Section 5 for problem descriptions.

	Problem A	Problem B	Problem C
Standard	29	22	0
Lexicase	30	29	12

near 0. Positive values of $M(p, \lambda)$ indicate problem modality, with a value of 1 indicating maximum modality. Note that any problem for which success is possible using lexicase selection, but impossible without it, will have $M(p, \lambda) = 1$.

Note also that this measure is comparative, and that it depends not only on the performance of the system with lexicase selection but also on the performance of the system *without* lexicase selection. In the example presented below, the non-lexicase system used ordinary tournament selection with a tournament size of 7. If we were to run the tests again with a different tournament size then the numbers produced for $M(p, \lambda)$ would probably be different, but we would expect to see similar orderings and trends. That is, if for problems p_1 and p_2 such that $M(p_1, \lambda) > M(p_2, \lambda)$ with one reasonable λ, then we would also expect this to hold for other reasonable choices of λ. If this is true then the measure would have some utility in spite of its flaws.

5. ILLUSTRATIVE EXAMPLE

Consider an integer symbolic regression problem involving thirty x (input) values ranging from -15 to 14 and y (target output) values produced by the equation $y = x^2 + x + 1$. This will be our illustrative nonmodal problem, to which we will refer as Problem A. Next consider the problem that is identical to Problem A for its first fifteen x values, but for which the y values for the remaining fifteen values are produced by the equation $y = 7x$. This problem, which we will call Problem B, appears to require two modes of response. Finally, consider the problem that uses $y = x^2 + x + 1$ for the first ten values, $y = 7x$ for the next ten, and $y = 3x^2 - 3$ for the final ten. This final problem, which we will call Problem C, appears to require three different modes of response.

We used GP to solve these problems with a function set containing: `+` (a 2-argument addition function); `-` (a 2-argument subtraction function); `*` (a 2-argument multiplication function); `/` (a 2-argument protected integer division function, with which division by zero yields zero); `ifz` (a 3-argument conditional function that returns the value of its second argument if its first argument is zero, or the value of its third argument otherwise); `ifneg` (a 3-argument conditional function that returns the value of its second argument if its first argument is negative, or the value of its third argument otherwise); and `ifpos` (a 3-argument conditional function that returns the value of its second argument if its first argument is positive, or the value of its third argument otherwise). The terminal set included the symbol `x`, which will always be bound to the current fitness case's input value, and an ephemeral random constant that can produce integer values ranging from -5 to 4.

The baseline GP system used for these experiments is a very simple tree-based GP system written in under 200 lines of Clojure code.[6] Random program trees (for population initialization and for new subtrees in mutants) are generated recursively by choosing to add, at each step for which the depth limit has not yet been reached, a function or a terminal each with the probability 0.5; only terminals are selected when the depth limit has been reached. The population size is $1,000$ and the program trees in the initial population are generated with a depth limit of 5. Parents are selected (in the non-lexicase condition) using tournaments of size 7. Offspring are produced by: subtree mutation, in which a randomly selected subtree is replaced with a newly generated subtree of maximum depth 3; crossover, in which a randomly selected subtree is replaced with a subtree that is randomly selected from another parent; or straight reproduction. The selection of subtrees for mutation and crossover is uniform, with all subtrees having equal probability of being selected. No size limits are imposed on the products of mutation or crossover, so it is possible for trees to grow without bound. A child is generated by mutation with probability 0.49, by crossover with probability 0.49, and by straight reproduction with probability 0.02. Each run is permitted to continue for 101 generations or until a solution is found, whichever comes first.

Table 3: Problem modalities computed from differential performance of lexicase selection, shown in Table 2, using Equation 1. See Section 5 for problem descriptions.

	Problem A	Problem B	Problem C
Modality	0.017	0.137	1

Table 4: Mean final best fitnesses (where fitness is error, so lower is better) for standard GP and GP with lexicase selection on problems with three levels of modality (from 30 runs per condition). See Section 5 for problem descriptions.

	Problem A	Problem B	Problem C
Standard	0.033	3.833	49.776
Lexicase	0.0	0.049	11.511

We conducted 30 runs of this system for each of the three problems (A, B, and C) in the "standard" configuration, using ordinary tournament selection to select parents, and another 30 runs for each problem using lexicase selection (specifically "global pool, uniform random sequence, elitist lexicase parent selection" as described above).

The results, in terms of the number of runs that successfully produced solutions, are shown in Table 2. Table 3 shows the resulting modalities for the three problems, calculated according to Equation 1. Table 4 shows the mean final best fitnesses; that is, the averages, across all runs in a condition, of the best (lowest) fitness in the final generation of the run.

While these problems are small and artificial the results nonetheless serve to demonstrate several interesting phenomena. First, it is striking that the performance of the standard system degrades so rapidly and so completely as more modality is introduced, with no runs solving Problem C. Second, lexicase selection does perform much better, with a widening performance gap as problem modality is

[6]Clojure is a dialect of Lisp; see http://clojure.org.

increased. Finally, the modality measures computed from the success data appear to be quite reasonable, with Problem A having a value near zero, Problem B showing modest modality, and Problem C appearing to be maximally modal.

Of course, a more sophisticated baseline GP system could certainly do better here, as could lexicase selection in conjunction with a more sophisticated baseline system. Our hypothesis, however, and our expectation, is that the pattern of improvement obtained by using lexicase selection will be similar, as will the computed measures of modality.

6. A CIRCULAR DEFINITION?

A critic might characterize the methodological move made in this paper with an imagined dialog such as the following:

Critic: You've developed a new problem-solving technique?
Author: Yes.
Critic: What's it good for?
Author: Modal problems.
Critic: How do you tell if a problem is modal?
Author: If my new technique works on it!

Clearly this appears at least somewhat problematic. How can it be justified?

For one thing, there does not appear to be any better definition of modality to which we can appeal at this time. As discussed in Section 4, several of the other obvious approaches fail to capture the essential notion of "having to do something qualitatively different in different circumstances."

A better justification may stem from an argument that the lexicase selection technique can actually be derived fairly directly by the assumptions underlying our notion of modality. We say that a problem is modal if a solution has to do something qualitatively different in different circumstances—that is, on inputs from different classes. But if the solution must do something qualitatively different in different circumstances then it would make little sense to compare the actions of individuals across those circumstances; it would only be reasonable to compare the actions of two individuals in exactly the same circumstances. We do not know in advance how many different kinds of circumstances there may be, or which fitness cases could be considered to fall together into the same kind of circumstance (requiring the same actions). But each individual fitness case counts as *some* kind of circumstance (possibly the same kind of circumstance as several other fitness cases), and individuals can therefore be meaningfully compared in terms of their performance on single cases. We also don't know, however, which cases are most important. So we conduct each comparison on the basis of a randomly ordered sequence of cases. Over many comparisons this allows for every case and every group of cases to serve as the basis for comparison, ensuring that we reward good performance in each circumstance, and on each group of circumstances, without ever requiring us to identify the circumstances or to rank them relative to each other.

If this line of reasoning is correct and the definition of modality can be seen to lead us directly to the lexicase selection algorithm, then it is indeed tautological that lexicase selection will help us to solve modal problems. But it is not necessarily tautological in the sense that we are left with circular definitions and nothing of value. Rather, it may be tautological while leaving us both with a good measure of modality and with a good way to solve modal problems.

7. CONCLUSIONS AND FUTURE WORK

As noted in the paper's title, this is clearly preliminary work. We have outlined a set of ideas and illustrated something of their potential, but the small example presented in Section 5 does not establish either the utility of lexicase selection or the utility of the measure of modality based on lexicase selection performance; it serves only to *illustrate* these concepts. Many more runs, on many more problems, would be required to demonstrate utility empirically. Indeed, such runs might instead undermine the case for the measure's utility, if, for example, Equation 1 yields a high value for some problem that does not appear to be truly modal. Only further experimentation will tell us whether or not this will happen, and even if experiments *do* demonstrate utility one would want to conduct further experiments to see how other existing methods, such as the multi-objective methods mentioned in Section 2, compare to the methods presented here.

In addition to collecting empirical evidence, one might also seek to develop mathematical models and analysis that support predictions of the utility of lexicase selection and the proposed measure of modality. We have not yet provided such models or analysis either.

That said, we believe that we can nonetheless draw some (preliminary) conclusions and also chart some directions for future work that is likely to bear fruit. We have observed that modal problems are important and that there are gaps in our knowledge about how to solve them with GP. We have seen a new approach to solving modal problems using a new, simple parent selection algorithm—lexicase selection—and we have seen how a metric for problem modality can be defined in terms of the performance improvement provided by lexicase selection.

Although this paper provides only a minimal, illustrative example, we can report anecdotally that we have also used lexicase selection on a version of the geometric formula problem described in Section 1, and that it succeeded while all other methods that we tried failed. Also anecdotally, we can report that success on Problem C in Section 5 appears to be due specifically to lexicase selection and not just to higher selectivity; runs with larger traditional tournaments also generally fail. So we have reason to believe that lexicase selection has real utility, even though this paper provides little firm evidence for this belief. We do hope, however, that this paper has made it clear how such evidence could be obtained, and that it has provided the motivation for doing so.

One area in which work should be done is to further delineate the situations in which lexicase selection is likely to be useful, along with situations in which it might founder. An assumption made in the development of the ideas presented here, and embodied in the examples on which they have been tried, is that the size of the GP population will be larger than the number of fitness cases. This makes it likely that each fitness case will appear in the first position for at least one and probably many more selection events, ensuring that good performance on each case is actually rewarded via contributions to the following generation. But if there is a very large number of fitness cases—as there might be, for example, if we are evolving end-user software applications and deriving our fitness cases from a large suite of software tests—then adjustments to the algorithm may be necessary. If so, then we might start by considering alternatives to the use of a global pool in step 1a of Figure 1, or alternatives to uniformly random sequences in step 1b, or alternatives to elitism in step 2a. Some of these alternatives may be worthwhile in other situations as well. For example, with the elitist version of lexicase selection presented here, if there is one individual that is elite on all cases then it will be selected in *all* selection events. This might have disastrous consequences for population diversity, and so alternatives should be considered if it appears that this is preventing the solution of a particular problem. It would also be interesting to examine the use of lexicase selection on problems that involve noisy data; one might expect this to require relaxation of the elitism in step 2a, although further study would be required to support or undermine this intuition.

On the other hand, it is possible that lexicase selection may provide benefits beyond those that were hypothesized above. For example, prior work has developed ways to reward individuals that do well on fitness cases that appear to be particularly hard to solve, where hardness is determined by looking at how well a case has been solved over the preceding history of the GP run [3]. That technique was not developed explicitly for modal problems, but lexicase selection appears to provide a similar effect automatically, as a side effect of the way in which it traverses the population during selection.

Both lexicase selection and the modality measure proposed here may also have utility outside of GP, for example in other areas of evolutionary computing. It should be relatively straightforward to apply these ideas to any system that performs selection based on fitness tests that involve multiple fitness cases.

In any event it is clear that there is a lot to be done, and we hope that this paper has made the case that the ideas presented here are worth exploring further.

8. ACKNOWLEDGMENTS

Thanks are due to the workshop organizers and anonymous reviewers for their open-mindedness and excellent suggestions, to Thomas Helmuth, Omri Bernstein, Kyle Harrington, Emma Tosch, Kwaku Yeboah Antwi, and Rebecca S. Neimark for discussions related to this work and comments on drafts, to Josiah Erikson for systems support, and to Hampshire College for support for the Hampshire College Institute for Computational Intelligence. This material is based upon work supported by the National Science Foundation under Grant No. 1017817. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] K. A. Benson. Evolving automatic target detection algorithms that logically combine decision spaces. In *Proceedings of the 11th British Machine Vision Conference*, pages 685–694, Bristol, UK, 2000.

[2] K. Imamura, T. Soule, R. B. Heckendorn, and J. A. Foster. Behavioral diversity and a probabilistically optimal GP ensemble. *Genetic Programming and Evolvable Machines*, 4(3):235–253, Sept. 2003.

[3] 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*, pages 61–75. Springer, 2008.

[4] M. Kotanchek, G. Smits, and E. Vladislavleva. Pursuing the pareto paradigm tournaments, algorithm variations & ordinal optimization. In R. L. Riolo, T. Soule, and B. Worzel, editors, *Genetic Programming Theory and Practice IV*, pages 167–186. Springer, 2006.

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

[6] W. B. Langdon. *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!*, volume 1 of *Genetic Programming*. Kluwer, Boston, 1998.

[7] W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 66–73. Morgan Kaufmann, 2001.

[8] S. Luke and L. Panait. Lexicographic parsimony pressure. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836. Morgan Kaufmann Publishers, 9-13 July 2002.

[9] L. Panait and S. Luke. Alternative bloat control methods. In *Genetic and Evolutionary Computation – GECCO-2004, Part II*, volume 3103, pages 630–641. Springer-Verlag, 2004.

[10] 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).

[11] L. Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046–4072, 2009.

[12] F. Schmiedle, N. Drechsler, D. Grosse, and R. Drechsler. Priorities in multi-objective optimization for genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 129–136. Morgan Kaufmann, 2001.

[13] T. Soule, R. B. Heckendorn, B. Dyre, and R. Lew. Ensemble classifiers: Adaboost and orthogonal evolution of teams. In R. Riolo, T. McConaghy, and E. Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, pages 55–69. Springer, 2010.

[14] L. Spector and J. Klein. Trivial geography in genetic programming. In T. Yu, R. L. Riolo, and B. Worzel, editors, *Genetic Programming Theory and Practice III*, pages 109–123. Springer, Ann Arbor, 2005.

[15] L. Vanneschi, M. Tomassini, M. Clergue, and P. Collard. Difficulty of unimodal and multimodal landscapes in genetic programming. In *Genetic and Evolutionary Computation – GECCO-2003*, pages 1788–1799. Springer-Verlag, 2003.