

Evaluation of Evolutionary Data Mining Techniques

Thomas Fernandez, MSCS, ABD

Instructor, Florida Atlantic University, Boca Raton, FL, USA

Abstract

To successfully develop new evolutionary data mining techniques, we need a methodology for evaluating their benefits. Evolutionary techniques are stochastic so experiments must be designed to report the results with a high level of statistical confidence. Clearly the evaluation of evolutionary data mining techniques must be based on the performance of the resulting classifier systems using observations from outside of the training data set. These experimental results should not be specific to a given database. The experiments should be repeatable using a variety of databases representing problems of different difficulty. During the experiments each technique being compared must be allocated the same amount of computational resources even if the resources are provided by multiple and dissimilar computers.

In our development and testing of new evolutionary data mining techniques we have implemented a Genetic Programming system called NUGP designed for the statistical comparison of Genetic Programming techniques. To ensure the fairness of our experiments, we propose a new metric, the Number of S-expression Nodes Evaluated, to measure computational effort in Genetic Programming systems. We also implement a program called MAKDAT for generating sets of synthetic data with real world properties and different levels of difficulty.

Keywords

Evolutionary Data Mining Genetic Programming

1. Introduction

Genetic Programming (GP) is a process of evolving computer programs based on Darwinian evolution. GP starts by creating a population of random computer programs. Individuals in this population are allowed to mate with each other resulting in offspring which have combined properties of both the parents. This process of mating is combined with random mutation to constantly create new programs. All of the programs are subjected to a fitness function that evaluates how well they perform their intended task. Programs with lower fitness are removed from the population. The pressure of favoring the more fit individuals causes the population to evolve. This continues until a program is found that is considered to be a solution to the problem, or until the provided computational resources are exhausted and GP fails to find a solution (Koza,, 1992).

GP can be used to discover the relationships between independent variables and dependent variables and express these relationships in the form of computer programs. The computer programs generated by GP can then be used on new sets of data to make predictions about unknown dependent variable values. The ability to predict or classify independent variables, which have complex relationships with sets of

independent variables, is essential for almost all modern engineering, science and social science.

Many techniques already exist that allow us to predict dependent variable values. All of them tend to make assumptions about the final form of the solution and then find parameters that optimize the solution within their predefined form. Examples of this are linear, nonlinear, polynomial and multivariate regression and neural networks. GP is a unique method for symbolic regression because it searches the space of all computable functions for the optimal form as well as the parameters of the solution. GP is thus extremely flexible and can be used for a wide variety of novel problems where the form of the solution is unknown.

GP is however computationally intensive. Evolutionary computation in general is often an inefficient process requiring enormous amounts of computational resources. Even then it is not guaranteed to provide a solution of high quality. It is therefore essential that we experiment with the process of GP and attempt to discover new GP techniques that will allow GP to have a higher probability of finding better solutions while using less computational resources.

As we experiment with different GP techniques we need a methodology for determining if one technique is an improvement over another. In this paper we will describe such a methodology. The case study that we will use for this is a comparison of what we call Regular GP with a modified GP algorithm that we will call Locality GP.

We have a theoretical basis for believing that Locality GP will work well. In Regular GP all of the individuals in the population are typically subjected to the same fitness function but in nature individuals in a biological population are spread over multiple geographical localities that have climatic and other environmental differences. Each locality effectively has a different fitness function for determining optimal survival traits. Charles Darwin observed this in the Galapagos Islands, where different environments caused the evolution of different beak shapes in finches (Darwin, 1909).

Some species (like *Homo sapiens*) do not become specialized to local ecosystems, instead they evolve generalized and adaptable solutions to the problem of survival. These species are considered to be generalists and are able to survive in a wider range of habitats, have larger populations and are less susceptible to extinction (Farrell et al., 2000). When GP is used for symbolic regression or classification we also want to evolve robust generalist solutions that are not specific to the data that was used for training. It is essential that the evolved solutions can make good predictions or classifications for observations outside of their training set.

To encourage the evolution of these generalized solution in Locality GP we have divided the population into Local Environments or *Localities* which are contiguous overlapping sub-populations defined as all the individuals in a GP population array starting at a specified start index and going up to but not including the individual at a specified end index. Each Locality has it's own Localized Fitness Function (LFF) based on a unique subset of the training observations that is assigned to that Locality. The LFF calculates the average absolute prediction errors only for the dependent variables in the Locality's subset of training observations.

The offspring from the mating process can end up in neighboring Localities. Some evolved solutions may be specific to only their own Locality's training observations. The offspring of these individuals will probably not survive in neighboring localities with LFF's that use different sets of training observations. On the other hand solutions which have evolved to correctly represent an underlying model of all the training data will have the potential to spread their genetic material past neighboring localities and then throughout the population.

Thus we have a theoretical basis for believing that Locality GP will evolve programs that are more accurate when applied to observations outside the training data. It is important that we also determine empirically that this is the case. We can do this by applying statistical methods to compare the results of Regular GP with Locality GP when both are applied to a variety of problems in a controlled manner.

2. The Test System

The Genetic Programming System we have developed for these experiments is called NUGP. It is a classical Koza type GP system (Koza, 1992) designed as an experimental test bed for new GP techniques. It includes a Statistical Analyzer that determine if a technique is providing improvements to GP by comparing values like execution time, training scores, test scores, and tree size to see if there are significant statistical differences between different sets of GP runs (Jain, 1991)(Mendenhall and Ott, 1980).

The NUGP system runs in a multi-processor environment. Twenty PC's running MS Windows XP and networked with a 10 Mbit LAN were used to conduct the experiments for this research. A primary design concern for NUGP was speed, so that experiments could be done efficiently and with sufficient GP runs to make experimental conclusions with a high degree of statistical confidence. Much of the systems speed was achieved by optimizing the utilization of primary and secondary memory caches by having data representations of the S-expressions that are compact and contiguous.

3. The Test Data

Ultimately we intend to use NUGP for regression analysis, prediction and classification with difficult real world problems. in the fields of engineering, science and social science. Real world problems are often hard because their general form is unknown and because there are random, chaotic and hidden factors. Real world problems often have missing data and significant measurement errors. In spite of this ultimate goal real world problems may not be effective for testing improvements to GP. Because of the expense of collecting real world data we are often limited in the number of similar but independent data sets that we can use for our experiments. If we use limited numbers of real world data sets to test potential improvements to GP we run the risk of simply fitting the methodologies to the data. Therefore we have chosen to use synthetic data sets to test the potential improvements of Locality GP.

We can easily generate many synthetic data sets with similar properties thus facilitating statistical analysis. We have control over some of the properties in the synthetic data and can learn what properties make data sets hard for GP to model. We can create data sets of varying degrees of difficulty and can test how well GP systems work with a range of different problems.

3.1 Synthetic Test Data Generation

To build these synthetic data sets we have developed a program called MAKDAT. It creates text files with data sets that are organized into columns and rows. Each row contains a single observation which is a set of independent variable values and an associated dependent variable value. Each column represents an independent variable except for the last column which is the dependent variable and the first column which is just a sequential numbering of the synthetic observations. The values for each independent variables in each observation are generated randomly using gaussian distributions with randomly assigned means and standard deviations. The dependent variable is calculated using a randomly generated S-expression.

The leaves of the S-expression are numeric constants and the independent variable values. The interior nodes are the numerical operators of addition, subtraction, multiplication and division. This allows the creation of dependent variables with linear and non-linear relationships to the independent variables.

There is also a conditional branching operator called *IfGreaterThanZero* which can also be used in the interior of MAKDAT's S-expressions. This operator takes three arguments. It returns the value of the second argument if the first argument is greater than zero and otherwise returns the third argument. This operator is important because it allows the creation of non-continuous functions and creates dependent variables that are effectively generated by a combination of numeric operations and a decision tree. However MAKDAT is not limited by this. It is easy to add additional operators to the MAKDAT S-expressions. It has been shown in GP that adding operators like indexed memory make S-expressions Turing complete (Teller 1994). Adding such operators to MAKDAT would allow it to randomly select functions, for generating dependent variable values, from the space of all computable functions. In this way it could generate data sets with relationships, between the independent and the dependent variables, that are extremely hard to discover.

The difficulty of the data sets generated by MAKDAT can be adjusted with its input parameters. There are four ways that the user can increase the difficulty of the data sets. Larger sets of independent variables, larger numbers of independent variables that do not contribute to the dependent variable value, larger amounts of injected synthetic measurement errors and larger (more complicated) S-expressions.

Table 1: Data Set Generator Parameters

Parameter	Easy	Medium	Hard
Independent Variables	5	10	20
Used Independent Variables	5	5	10
Tree Depth	4	5	7
Added Error	0.0	0.001	0.001

Three different types of data sets were generated for evaluating and comparing the performance of Locality GP and Non-locality GP. We refer to these three types of data sets as *Easy*, *Medium*, and *Hard*. The parameters used to generate each of these is shown in in Table 1.

For each of our experiments we use the MAKDAT program to generate 7000 observations. 5000 of these observations are used for training and the remainder is divided into two sets of 1000 observations that are used as the test data.

The data files that have the values of the the independent and dependent variables for the observations in the training and test data sets for all three problems can be downloaded from our web site at www.cse.fau.edu/~thomas. Documentation of the simple text file formats is also available along with the raw output files from this experiment. Collaboration with other researchers who would like to use these or other synthetic data sets is welcome.

4. Evaluation Criteria

4.1 Measuring Computational Effort in GP

Often computational effort in GP is measured in generations (Koza, 1992) or in execution time (Tokui and Iba, 1999). We have chosen the *the number of S-expressions nodes evaluated* as our measure of computational effort. It fairly compares the effort expended evaluating S-expressions of different sizes and is independent of hardware even when NUGP runs on parallel systems. It is also independent of software implementation and thus facilitates the replication and objective comparisons of experiments by

other researchers. The *number of S-expression nodes evaluated* is usually quite large so we actually use the term MNE to stand for Million Nodes Evaluated¹.

4.2 Determining Best Test Results

After allocating the same number of MNE's to multiple GP systems how can we determine which one is better? In our case we are evaluating GP systems that ultimately will be used for decision making with real-world problems like financial analysis. In these situations accurate prediction of the dependent variable is not the real goal. The goal is to have the highest possible percentage of correct decisions. For example the decision to buy a financial instrument could be based on a prediction of the future price. If the price does go up than the decision is considered correct even if the actual future price is different from the prediction.

After reaching a predetermined number of MNE's the GP runs are stopped. At that point the best individual is used to make predictions of the dependent variables in two independent sets of test data. Each prediction is considered to be a correct decision in the following three cases.

- (1) If the prediction and the dependent variable are both above the average of the dependent variable values.
- (2) If the prediction and the dependent variable are both below the average of the dependent variable values.
- (3) If the prediction and the dependent variable are both close to the average of the dependent variable values.

4.3 Calculating Average Dependent Variable Values

In the three correct decision criteria described above the *average dependent variable value* is the average of the dependent variable values from the 5000 training observations. (These are the only observations that can be considered available before testing the best individual in the population.) In the third criteria for correctness we define *close to the average* as a range of values that extends equally above and below the average dependent variable value such that 10% of the 5000 training observations fall within that range.

By allowing predictions that are close to the average to count as correct we are being realistic about real-world problems where there might be situations that are too close to call and GP is allowed the option of deferring the decision to a human expert. There might also be domains such as financial analysis where there is usually a decision to go long or short depending on whether the predicted future value of a financial instrument is higher or lower than its current market value. There might however be situations where it might be best to go neither long or short (referred to as banking) when the GP system is either unsure of it's prediction or when the predicted change in the price is less than the transaction costs.

4.4 Calculating Test Scores

Using the predictions for the two test data sets and the three criteria described above we calculate the test score for each of the two test data sets. The test score is the fraction of the test observations that were misclassified as above average, close to average, or below average. This is calculated by dividing the number of incorrect classifications by the total number of observations resulting in a numeric value

¹ MNE is a unit of measurement that is independent of hardware and software implementation but the MNE's *per minute* can be used as a benchmark to measure the relative performance of different hardware and software implementations. For example NUGP is currently running around 750 MNE's per minute on a 2.8 Ghz Pentium IV with 1.5 Gigabytes of Ram and around 300 MNE's per minute on a 1.8 Ghz Pentium IV with 256 Megabytes of Ram.

between 1.0 (if every prediction is considered wrong) and 0.0 (if they are all considered to be right). Note that GP systems with lower test scores are better. Also note that if the test scores for the two data sets are not similar then we would be concerned about their validity but in all our experiments the two test scores were always similar.

5. Methodology

5.1 Implementation of Regular GP

Before we examine the methodology for Locality GP we need to review our implementation of Non-Locality or Regular GP.

NUGP runs asynchronously. Individuals with better fitness are continuously selected for mating using tournament selection and their offspring continuously replace individuals which are determined to have lower fitness. In our experiments we always selected two tournament participants. The winner is the individual with the better fitness score and is designated as the first parent. The loser is removed from the population to make room for one of the two children of the mating process. This process is repeated to find a second parent and another individual that will be replaced by the second offspring. The selected parents and their new offspring remain in the population and are available for future tournaments.

The mating operator is implemented as classic ``Koza" style mating (Koza, 1992) where a crossover node is randomly selected in each of the parent's S-expression trees and then these nodes (and their subtrees) are exchanged.

There is also a mutation operator, which is applied to random individuals without regard for their fitness. The mutation operator is also implemented in a classic ``Koza" style (Koza, 1992). An individual node is randomly selected as a mutation point. The selected node (and its subtree) is replaced with a new randomly generated subtree. This process is repeated continuously until a specified number of MNE's is reached. At that time the best individual so far is considered to be the result of the GP run.

That best individual is then used to predict the dependent variables in the two sets of test observations that were not used during training. Using the evaluation criterion described above the best individual will receive two test error scores, which are the fraction of test predictions that were considered incorrect.

5.2 Implementation of Locality GP

NUGP can run as a Regular GP system as described above. By setting parameters in a control file NUGP can also be run as a Locality GP system.

Locality GP is implemented in a similar way to Regular GP except that three parameters are used to limit the selection of the parents to a specific section of the population array. The same parameters are also used to specify which training observations are used during each call to the fitness function.

These three parameters are *LocalityStartIndex*, *LocalitySize* and the *LocalityCircleIncrement*. At the beginning of each round^{footnote{NUGP is an asynchronous system and so does not have generations, but a round is roughly analogous to a generation in a synchronous system. During each round n individuals will mate, where n is the size of the population. *LocalityStartIndex* is set to a randomly selected integer between 0 and the population size minus one. The first Locality for the round is all the individuals from the *LocalityStartIndex* up to but not including the individual at *LocalityStartIndex* plus *LocalitySize*.}

Locality GP differs from Regular GP by using a Localized Fitness Functions (LFF). With Regular GP the fitness function uses the same training observations every time it is called, but with Locality GP different subsets of the training observations are used by the LFF depending on the current Locality. This is implemented by assigning each individual in the population array to an associated training observation. Each time the fitness function is called all of the training observations associated with the individuals in the current Locality are used to evaluate the fitness score. Another difference is that Locality GP uses LS to pick candidates for the tournaments only from a given Locality and not from the entire population like Regular GP.

After each mating the current Locality is moved in the population array. This is done by adding the *LocalityCircleIncrement* to the *LocalityStartIndex* to make a new *LocalityStartIndex*. The new *LocalityStartIndex* marks the beginning of the next Locality which includes the next *LocalitySize* individuals. If the sum of the *LocalityStartIndex* and the *LocalitySize* is larger than the Population array size then the new *LocalityStartIndex* will be wrapped around to the beginning of the toroid population array. Likewise if a Locality extends past the end of the population array it will also be wrapped around to include individuals from the beginning of the array.

For all of our experiments we've used a *LocalitySize* of 100 and the *LocalityCircleIncrement* of 75 this means that each current Locality overlaps the previous Locality by 25 individuals and also overlaps the next Locality by 25 individuals. It is important that the Locality size be larger than the Locality circle increment so that there will be an overlap between adjacent localities. This overlapping of Localities is the mechanism that allows the transfer of genetic material to adjacent Localities and the eventual spread of successful genetic material throughout the entire population.

5.3 The Fitness Function and GP Parameters

GP was trained using the following fitness function:

$$fitness = w_e \left(\frac{\sum_{i=1}^n (2^{E_i} - 1)}{n} \right) + w_{ne} \left(\frac{\sum_{i=1}^n E_i}{n} \right) \quad (1)$$

E_i is the prediction error for observation i defined as

$$E_i = |y_i - \hat{y}_i| \quad (2)$$

Where y_i and $\hat{y}_i$ are the value of dependent variable value and the prediction of the dependent variable for observation i . n is the number of observations used by the fitness function.

The fitness function in formula (1) can be thought of as a hybrid of two fitness functions. The first is an Average Exponential Error (AEE) fitness functions where 2 is raised to the power of the error of each observation. Using this fitness function, the system is driven to correct large errors first and ignore smaller errors. The reason for subtracting one from this exponential error term is so that zero will signify no error, just as it does with the Average Absolute Error (AAE) fitness function.

The second part of the hybrid fitness function is simply a standard Average Absolute Error(AAE). It places the same importance on correcting observations with large or small errors. The two components of the hybrid fitness function AEE and AAE are weighted using w_e as the exponential weighting and w_{ne} as

the non-exponential weighting. The values for w_e and w_{ne} are set by the user of NUGP as parameters in control files. Typically w_e and w_{ne} would add up to one. The exponential error values are initially much larger than the error values themselves. As the errors values become smaller the opposite is true. Thus early on in the evolutionary process GP is encourage to correct predictions for the observations with large errors. Later when the errors are smaller, GP is encouraged to work more evenly on all the observations used. In our experiments the values of w_e and w_{ne} were both set to 0.5.

In each round mutation is applied to 2% of the individuals randomly selected with a uniform distribution. Numeric Mutation was also applied to 10% of the numeric constants in 10% of the population. Numeric Mutation replaces the constants with a randomly generated value using a normal distribution around the previous value with a standard deviation equal to an individual's fitness score divided by ten (Fernandez and Evett, 1998). S-expression trees were limited to 1024 nodes. Initial populations were generated with sparse and bushy trees with depths between 3 and 6 using ramped half and half tree generation (Koza, 1992). All of these parameters were held constant for all runs. All runs in the experiment with the *Easy* problem were allowed to continue for 50 thousand MNE's (50 billion S-expression nodes evaluated). In the experiment with the *Medium* problem runs continued for 100 thousand MNE's and for the *Hard* problem 150 thousand MNE's were used.

The GP function set includes addition, subtraction, multiplication, protected division, and a function called *IfGreaterThanOrEqualToZero* described previously. The terminal set included the independent variables and the random ephemeral constant (Koza, 1992).

6. The Experimental Design

Our hypothesis is that Locality GP produces better solutions than Regular GP given the same amount of computational effort. We define better solutions as ones that have fewer errors when classifying observations in test data sets as either above average, near average or below average. We define two GP systems to have the same amount of computational effort, when they have both had the same number of MNE's. To test our hypothesis we have designed experiments where 30 Locality GP runs are made and another 30 Regular GP runs are made. The best individuals from each run are tested using a test data set. For both the 30 Locality GP runs and the 30 Regular GP runs we calculate an average of the fraction of test cases that are classified incorrectly. These averages are then compared to determine whether Locality GP or Regular GP is more accurate at classifying the test observations. The sample variances are considered and a z statistic is calculated. We examine the absolute value of the z statistic to see if it is greater than 2.576 (required to reject the Null hypothesis with 99% confidence in a two tailed test).

We examine the benefits of Locality GP with three experiments using problems of different difficulty which we have referred to as *Easy*, *Medium*, and *Hard*. For each of these problems an experiment was run with thirty GP runs with Locality GP and another thirty with Regular GP. In each experiment the only difference between the two groups of runs was that Locality GP was used with one and Regular GP was used with the other. To further validate each of our experiments, we report the average test scores from a second set of test observations independent of the first test data set. We verify that there are similar test scores from both test data sets.

7. The Experimental Results

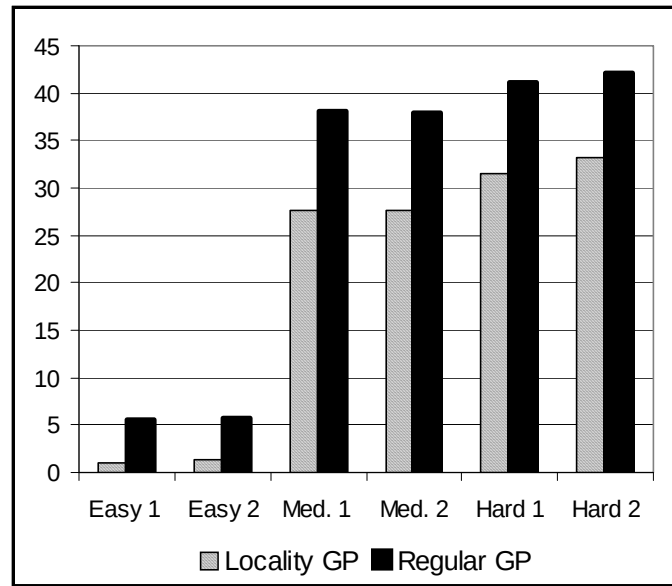


Figure 1: Average percent misclassified in test data sets with Locality GP and Regular GP

In Figure 1 we see that the average fraction of misclassified observations for all three problems (in both test sets) was smaller with Locality GP than it was for Regular GP. Note that bar in the graph represents the average of 30 runs.

Table 2: Statistics for the three problems with Locality and Regular GP

Easy Problem			Medium Problem		Hard Problem	
Locality GP	Test 1	Test 2	Test 1	Test 2	Test 1	Test 2
Ave. Misclassified	0.01060	0.01280	0.27660	0.27577	0.31507	0.33147
Variance	0.00024	0.00026	0.03866	0.03913	0.00204	0.00211
Regular GP	Test 1	Test 2	Test 1	Test 2	Test 1	Test 2
Ave. Misclassified	0.05570	0.05787	0.38093	0.37847	0.41207	0.42073
Variance	0.00075	0.00078	0.00991	0.01023	0.00126	0.00120
z Statistic	-7.86770	-7.67531	-2.59489	-2.53150	-9.24126	-8.49920

We consider the sample variances and calculate the z statistic. This data is summarized in Table 2. We analyze these results using a *Two Tailed Large Sample Comparison of Population Means* (Mendenhall and Ott 1980).

In one case with the Medium problem and the second set of test data the z statistic is -2.531496. The magnitude of this z statistic was below 2.576 and so we can not reject the null hypothesis with a confidence of 99%. The z statistic is still greater in magnitude than 1.960, so in this case we still have a 95% confidence in the superiority of Locality GP over Regular GP. In all of the other cases the absolute value of the z statistic is greater than the test statistic 2.576, and we have a 99% confidence in the superiority of Locality GP.

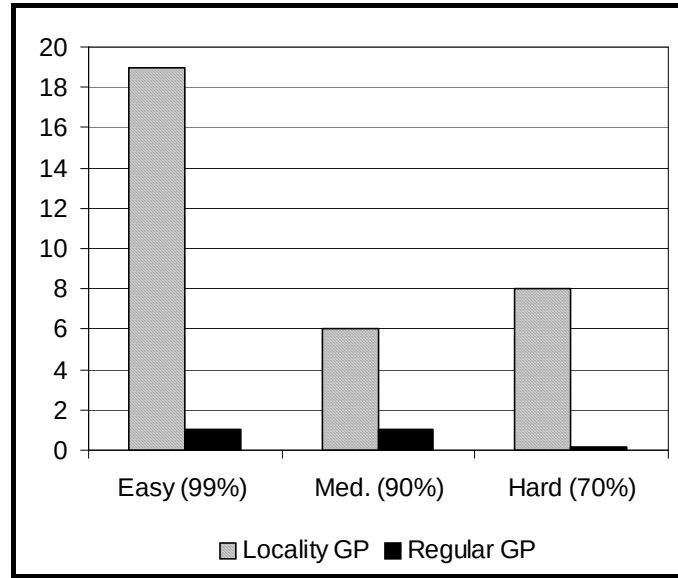


Figure 2: Number of GP runs out of 30 that evolved solutions that reached a given percentage of classification accuracy for both test data sets.

Often GP runs are repeated until certain criteria are met (Koza, 1992). For this reason we look at how many times out of 30 runs a solution reached a specific threshold of accuracy with both test data sets. For example, 19 out of 30 Locality GP runs produced models that were at least 99% accurate in classifying the observations in both test data sets with the *Easy* problem, but only 1 out of 30 Regular GP runs was able to reach this level of accuracy. For the *Medium* problem 6 out of 30 Locality GP runs was able to reach at least 90% accuracy for both test data sets and only 1 out of 30 Regular GP runs achieved this. Finally for the *Hard* problem 8 out of 30 GP runs was able to reach 70% accuracy on both test data sets, but none of the Regular GP runs was able to do this. These results are summarized in Figure 2.

8. Conclusions, Contributions and Future Work

We have run experiments with three different problems, which we have classified as *Easy*, *Medium* and *Hard* to compare a new algorithm for GP called Locality GP with Regular GP. In five out of the six statistical tests it is concluded, with a 99% confidence, that Locality GP is better than Regular GP in its ability to classify observations outside of the training data sets. In the one exception where the result could not be reported with 99% confidence we could still report it with a 95% confidence.

We have developed a tool called NUGP that can be used for classification and prediction and also allows the statistical comparison of different GP techniques. We have proposed a new measure of computational effort called MNE that is independent of software and hardware implementations. We have implemented a program called MAKDAT that builds synthetic data sets that have similarity to real world data for testing Locality GP. In the future we plan to apply Locality GP to real world symbolic regression and classification problems in the areas of engineering, science and social science.

References

- Baguette M. and Virgine S. (2003) *Local populations and metapopulations are both natural and operational categories*. Oikos 101(3):661-661.
- Darwin, C. (1909) The Voyage of the Beagle. http://www.infidels.org/library/historical/charles_darwin/voyage_of_beagle/index.shtml.
- Farrell B., Kelley S. and Mitton J. (2000) *Effects of specialization on genetic differentiation in sister species of bark beetles*. Heredity, 84(2):218-218.
- Fernandez T. and Evett M. (1998) *Numeric Mutation as an Improvement to Symbolic Regression in Genetic Programming*. Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, San Diego, California. pages 251-260 Springer-Verlag.
- Koza, J. (1992) *Genetic Programming: On the programming of computers by the means of natural selection*. MIT Press.
- Mendenhall W. and Ott L. (1980) *Understanding Statistics*. Duxbury Press.
- Menczer F., Degeratu M., Street N. (2000) *Efficient and Scalable Pareto Optimization by Evolutionary Local Selection Algorithms*. Evolutionary Computation, 8(2):233-247.
- Teller A. (1994) *Turing Completeness in the Language of Genetic Programming with Indexed Memory*. Proceedings of the 1994 IEEE World Congress on Computational Intelligence, Orlando Florida, Vol. 1, pages 136-141. IEEE Press.
- Tokui N. and Iba H. (1999) *Empirical and statistical analysis of genetic programming with linear genome*. System, Man and Cybernetics: IEEE International Conference Proceedings, Tokyo, Japan Vol 3. pages 610-615 IEEE Press.

Autorization and Disclaimer

Authors authorize LACCEI to publish the papers in the conference proceedings. Neither LACCEI nor the editors will be responsible either for the content or for the implications of what is expressed in the paper. If you have any questions, please contact:

Thomas Fernandez
Florida Atlantic University
tom@cse.fau.edu
561-297-3927
561-297-2800 Fax