Numeric Mutation Improves the Discovery of Numeric Constants in Genetic Programming

Matthew Evett
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida 33431
(561)297-3459; matt@cse.fau.edu

Thomas Fernandez
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida 33431
(561)297-2805; tfernand@cse.fau.edu

ABSTRACT

Genetic programming suffers difficulty in discovering useful numeric constants for the terminal nodes of its s-expression trees. In earlier work we postulated a solution to this problem called *numeric mutation*. Here, we provide empirical evidence to demonstrate that this method provides a statistically significant improvement in GP system performance on a variety of problems.

1 Introduction

A weakness of genetic programming (GP) is the difficulty it suffers in discovering useful numeric constants for the terminal nodes of the s-expression trees. At first glance, this weakness is somewhat surprising: Genetic Algorithms, from which GP is derived, are highly suited to the task of optimizing numeric parameters. GP's difficulty with numeric constants derives from their representation as tree nodes, while the reproduction operations (including mating and mutation) affect only the structure of the trees, not the composition of the nodes. Therefore the individual numeric constants are not affected by reproduction operations (including mutation and cross-over), and thus cannot benefit from them.

GP's difficulty with numeric constant generation is relatively well known. In a speech at GP97 John Koza said:

> The finding of numeric constants is a skeleton in the GP closet... [and an] area of research that requires more investigation.(Koza 1997)

The traditional way of generating new numeric constants is indirect, by combining existing numeric constants within novel arithmetic s-expressions. The leaves of the trees corresponding to these s-expression are all numeric constants, and so the s-expression necessarily evaluates to a numeric value. The entire s-expression can thus be viewed as a single numeric constant terminal node, with a value equal to that of the s-expression. We call this process of numeric constant creation *arithmetic combination*.

It is also possible to generate numeric constants even when none are provided in the original terminal set. For example, a terminal representing a variable could appear in an s-expression consisting of the variable being divided by itself, effectively yielding the constant 1.0. Once the constant 1.0 exists, 2.0 can evolve via an s-expression that adds 1.0 to itself. Having the constants 1.0 and 2.0, the constant of 0.5 can evolve via an s-expression that divides 1.0 by 2.0, etc. In this way the GP process can generate an arbitrary number of constants, even when no numeric constants are included in the original terminal sets. We call this process of numeric constant creation *arithmetic genesis*.

Although the spontaneous emergence of constants is possible via arithmetic genesis and arithmetic combination, the techniques are tedious and inefficient. We are examining several techniques for facilitating the creation of useful, novel numeric constants during a GP run. In this paper we report on one such technique, *numeric mutation*. We demonstrate that numeric mutation provides a significant improvement in the ability of GP to solve a symbolic regression problem.

2 History

Some of the early enhancements to the GP process facilitated the creation of constants. These enhancements (Koza 1992) consisted of including a small number of numeric constants and/or the *ephemeral random constant*, $\Re$, in the original terminal set. (Each time the ephemeral random constant is selected as a terminal in the creation of the population of generation 0, it is replaced with a randomly generated number within some specified range.)

Both of these techniques seed the genospecies with numeric constants. The ephemeral random constant is particularly helpful because it provides many different numeric constants in generation 0. Even so, most problems require a solution that uses numeric constants other than those provided in the original terminal set or generated by the ephemeral random constant. Such constants must be evolved tediously by arithmetic combination or arithmetic genesis (though the larger initial pool of numeric constants in the genospecies

does make arithmetic combination more likely.)

Even with the use of the ephemeral random constant and/or the presence of predefined constants in the terminal set GP still has difficulty generating sufficient numeric constants. In his first GP book (Koza 1992), John Koza uses GP on a problem consisting of discovering just a single numeric constant. Despite the use of the ephemeral random constant, the GP system still required 14 generations to create a solution, an s-expression comprising almost half a page. This is but one simple example, yet it illustrates that the creation of numeric constants remains a weak point of GP.

3 Numeric Mutation

Numeric mutation is a technique for facilitating the creation of useful, novel numeric constants during a GP run. Numeric mutation is a reproduction operation which, like mutation or cross-over, is applied to a portion of each population each generation. Numeric mutation replaces all of the numeric constants with new ones in the individuals to which it is applied. The new numeric constants are chosen at random from a uniform distribution within a specific selection range. The selection range for each numeric constant is specified as the old value of that constant plus or minus a *temperature factor*. The terminology derives from the similar concept of temperature in simulated annealing ((Kirkpatrick *et al.* 1983, Rumelhart and McClelland 1987) *et al*) in that when the temperature factor is larger, numeric mutation creates greater changes in the affected numeric constants.

The temperature factor is determined by multiplying the raw score of the best individual of the current generation by a user specified *temperature variance constant*, 0.02 here. The standardized fitness score of the best-of-generation individual approaches zero as it approaches a perfect solution to the problem domain. Consequently, the effect of this method for selecting the temperature factor is that when the best individual of a population is a relatively poor solution, the selection range is larger, and therefore there is an overall greater potential for change in the numeric constants of the individuals undergoing numeric mutation.

Over successive generations, the best-of-generation individual tends to improve and so the temperature factor becomes proportionally smaller. As the temperature factor decreases, numeric mutation causes successively smaller changes to the numeric constants. This should allow the GP process to "zero in on" (i.e., retain across generations with little change) those numeric constants that are useful in solving the given problem.

4 Experimental Evaluation

In our previous work (Fernandez and Evett 1998), we investigated the use of numeric mutation in only a single symbolic regression problem. Our research with numeric mutation is still at an early stage, but in this paper, we investigate the efficacy of numeric mutation in general. We show the efficacy of numeric mutation in a simpler symbolic regression problem and also in a much more difficult problem from the domain of financial analysis. The experimental hypothesis of our original experiment was that numeric mutation increases the effectiveness of the GP process in solving a single symbolic regression problem. The experiment involved the study of a problem, defined by 11 pairs of numbers representing the x and y coordinates of 11 points (*target points*) on a plane. Note that all of these points lie along the curve defined by the generating function:

$$y = x^3 - 0.3x^2 - 0.4x - 0.6 \quad (1)$$

This function is considered the target or goal of the symbolic regression only indirectly; an infinite number of curves pass through these 11 points, and the goal is to discover *any* function that passes within a distance of plus or minus 0.1 along the y–axis for the x value of each of the points.

At the end of each generation, the numeric mutation technique is applied to 40 randomly selected individuals of the 200 with the best fitness from a population of 1081. Each selected individual is replaced with a copy wherein each numeric constant has been mutated, as described in Section 3. The fitness function is reevaluated for each of the new individuals, so that the fitness-ranking of the population corresponds to the altered population.

The choice of the number of elements to be numerically mutated, the size of the group that they are selected from, and the use of 0.02 as the temperature variance constant, were based on experiments involving other regression problems that suggested that these values tended to maximize the benefit of the numeric mutation (Fernandez 1997).

To test our hypothesis, we ran our GP system 1000 times with numeric mutation and 1000 times without. Each generation of a numeric mutation run included the evaluation of the fitness function on 40 additional individuals (those created by the numeric mutation process). To compensate for the extra work done by the numeric mutation runs, the populations of the non-numeric mutation runs contained 40 more individuals. This makes comparisons between the results of the two techniques more fair, as both algorithms are then doing roughly the same amount of work. (Otherwise any performance advantage observed in the numeric mutation runs might be ascribed to the additional individuals evaluated therein.)

Each run was allowed to continue until a function was found that met the criterion described above, or until 50 generations were completed. Runs that discovered a function matching the target points within the 50 generation limit were considered successful. We ran our experiments on an AMD 166Mhz K6 running *Microsoft Windows 95*. We used our own hand-coded GP system (described in (Fernandez and Evett 1997*a*, Fernandez 1997)), using the control parameters specified in the tableau shown in Table 4 and an *elitist graduated overselection strategy* to select individuals from the population for reproduction and crossover, as described in (Evett and

Population size	1121 or 1081(NM)
% ramped complete growth	100
% ramped partial growth	0
Crossover Percentage	90
Mutation Pertcentage	0
Max Number of Runs	1000
Max Number of Generations	50
Max Nodes per Tree	200
Selection Strategy	Graduated Elitist
Initial Tree Minimum Depth	3
Initial Tree Maximum Depth	7
RandomSeed	0

Table 1 The GP tableau.

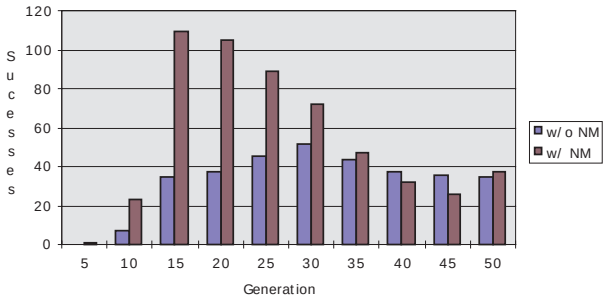

Figure 1 Number of runs that terminated successfully at each generation for GP using and not using numeric mutation.

Fernandez 1997).

4.1 Results

Of the 1000 runs without numeric mutation, 328 were successful, while 541 of the runs with numeric mutation were successful. Thus, runs using numeric mutation were about 65% more likely to terminate successfully than the plain runs. The success ratio of the GP system was clearly higher when using numeric mutation. To determine whether this outcome was statistically significant, we performed a Large-Sample Statistical Test for Comparing Two Binomial Proportions (as described in (Mendenhall and Lyman 1972), page 203).

The null hypothesis for the significance test was that the populations have the same success ratios, and the alternate hypothesis was that they were not the same. This choice of the alternate hypothesis necessitated the use of a two-tail test. It might be argued that numeric mutation is a modification to the GP technique involving additional work, and consequently we should be interested only if it provides an improvement to GP. An alternate hypothesis that reflects that argument is that GP with numeric mutation has a higher success ratio than GP without. Such a choice would permit the use a one-tailed test. We have used the first hypothesis and the corresponding two-tailed test because it is more stringent(Fogel 1997).

The results of the test were that we rejected the null hypothesis with 95% confidence. Thus we conclude that numeric mutation's improvement to GP is statistically significant for this problem. A further indication of this is that not only does numeric mutation yield successful runs more frequently, but also the successful runs require, on average, fewer generations than the successful runs on the GP system without numeric mutation. The average number of generations in a successful run with numeric mutation was 24.44, while the average without numeric mutation was 29.67. Figure 1 is a histogram of the number of generations required to complete the successful runs with and without numeric mutation. Each column of the graph corresponds to a sum across five generations. For example, the figure shows that of the 1000 runs using numeric mutation, 103 finished successfully between generations 21 and 25, inclusive.

This increase in efficiency was also reflected in run-time performance. The 1000 runs using numeric mutation required 8.28 hours to complete, while the 1000 runs without numeric mutation required 11.63 hours. The average time to complete a successful run with numeric mutation was 16 seconds while the average time without numeric mutation was 24 seconds. (The exact value of the timings is not important, but their relative values are. The timing information under Windows 95 is somewhat imprecise, but serves to support the conclusions derived from the other experimental results.)

The question still remains "Does numeric mutation provide a general benefit to the GP process, or is the benefit somehow limited to this spesific problem?". To investigate this issue we have conducted two similar experiments. The first is a simpler symbolic regression problem and the second is a much more difficult financial analysis problem.

4.2 A Simpler Regression Problem

The first of these additional problems is a simpler symbolic regression problem. Again we will try to evolve a function which passes through eleven given target points. This time the target points all lie on the curve defined by the function $y = x^2 + 3.141592654$. The problem is specified by providing 11 pairs of numbers representing the x and y coordinates of 11 points (the target points).

Again an infinite number of curves pass through these 11 points, and the goal is to discover any function that passes within a distance of plus or minus 0.00001 along the y-axis for the x value of each of the eleven target points.

4.3 Simpler Problem Results

Our experimental hypothesis is that the NM hybid improves GP's Ability to solve this simpler symbolic regression problem by having a higher success ratio then the non-hybrid GP.

We used a control set of GP runs where NM was not applied. We did 8 additional sets of 100 runs using different values for the Temperature Score Parameter (TSP). The population for these runs was set to 1041 so that the control population size of 1081 would be 40 individuals larger. The NM hybrid does 40 extra calls to the fitness function per generation to try to equalize the amount of work performed in all sets

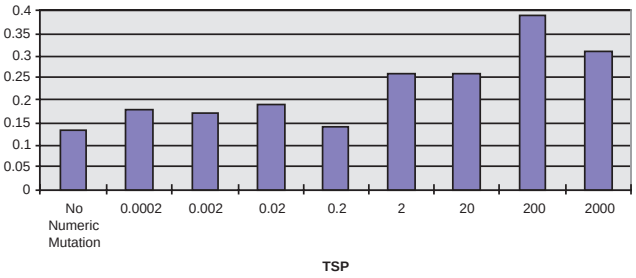

Figure 2 The success ratios of the control run and the NM runs when applied to the simple symbolic regression. problem

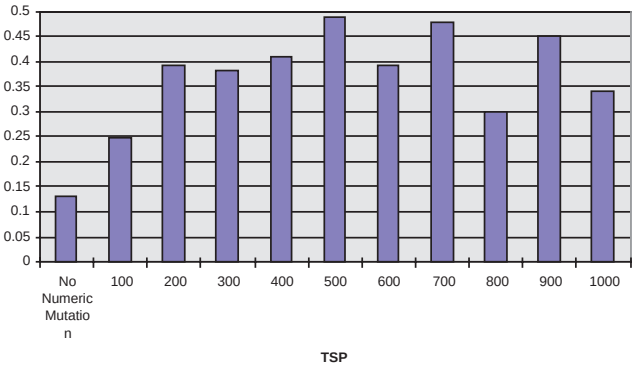

Figure 3 The success ratios of the control run (with no hill climbing) and additional NM runs when applied to the simple symbolic regression problem.

of runs. The equalization is not perfect, because the larger populations require more work during cross-over and selection, but this amount of work is small proportional to that spent on the individual evaluations.

Only the runs with a TSP of 2.0 or higher had success ratios with statistically significant differences from the control set. To gain more information about the effects of different TSP values on the success ratio of this technique we ran an experiment in which we varied the values of the TSP from 100 to 1000 in steps of 100. The results are shown in Figure 3.

All of these sets of runs had success ratios that had differences with the control set that were statistically significant.

The conclusion that we are led to from this is that in the domain of very simple symbolic regression that the use of Numeric Mutation provides a useful enhancement to the GP process, and that the choice of 500 is reasonable for the TSP.

We have shown that the results of our original experiment can be extended to simpler problems, but we still need to answer the more important question, "Can NM provide benefit when applied to harder problems?". In our previous work (Fernandez and Evett 1997*b*), we showed that GP could be applied to problems in the domain of financial analysis. We now use a problem from this domain to test the ability of NM to provide benefit when applied to more difficult problems.

4.4 A More Difficult Problem

Again the goal is to do a type of symbolic regression, in this case the target points are a financial time series. We use a target time series that is derived from the daily closing prices of the S&P 500 from the years 1994 and 1995. The S&P 500 is an index that is created by taking weighted averages of the stock prices of 500 large US companies. Instead of using just one independent variable as in the previous experiments, we use 33 independent variables taken from time series derived from the S&P 500 itself and the closing daily prices of 32 Fidelity Select Mutual Funds. All of the financial time series used for independent variables are preprocessed. The first step to transform each series into a series of 21-day moving averages. This reduces the effect of day to day fluctuations and allow the GP to concentrate on the general trend of these financial time series. The second preprocessing step converts the series resulting from the first preprocessing step into a series of percent changes between each day and 21 days prior. This normalizes the data within each series and between different series. The target data points are taken from the S&P 500 and preprocessed with the same two steps but are also shifted 10 days, so that the system is will try to predict the general trend, two weeks (10 business days) into the future.

Creating systems that predict the trends of a financial time series is a very important problem to the financial community. However it is not the goal of this research to create such a system, but rather to examine the potential benefit of NM as a technique for improving the GP's numeric constants when applied to a more difficult problem. This problem is significantly more difficult than the previous two symbolic regression problems. Besides having 33 independent variables, there is no predefined mathematical relationship between the independent variables and the target points. Because of the difficulty of the problem we have made the criteria for success much less stringent. The goal is to find a function that passes within 0.006 of the target value for 90% of the target points. A system for determining the econometric relationships between the independent variables and the target values would probably require more accuracy, but these less stringent termination criteria are often useful to prevent overfitting in complex domains. In any case, we emphasize again that our goal here is only to test NM as an enhancement to GP in this problem domain. A real system would also include testing on a set of data not used in training. This capability is included in the software we have developed but is not used here.

The top line in the graph in Figure 4 is the daily closing price of the S&P 500 (using the scale on the left). The solid line below it is the graph of the target time series after the preprocessing described above (using the scale on the right).

The dotted line in Figure 4 is a function evolved using standard GP (without NM) that would be considered a successful solution because it meets the criterion described above. The degree to which this curve differs from the target time series illustrates the latitude of our termination criteria.

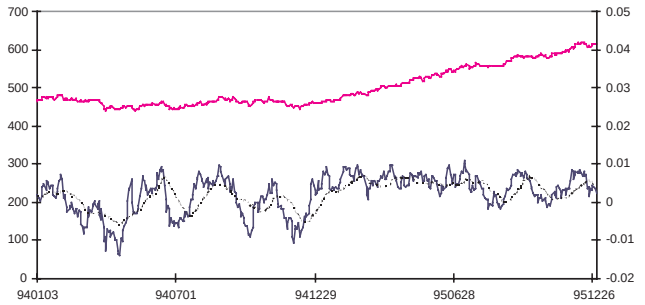

Figure 4 The S&P 500 (above) and a target financial time series derived from the S&P 500 (solid line on bottom) and an example of an evolved function (dotted line on bottom)

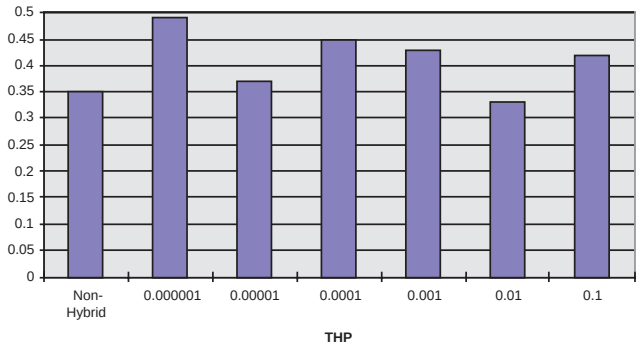

Figure 5 The success ratios of the control run (with no hill climbing) and the NM runs when applied to the financial domain problem.

The example evolved function was: $Y = (((0.38) - ((-0.20923) - (\text{FSPTX} - (((-0.79706)/(0.38)) * ((\text{FSUTX} - \text{FSCSX}) * (\text{FSCGX} - (-0.34247)))))) * (\text{SPX} * ((0.82794)/(0.54431))))$ The independent variables of this evolved function represent various time series. (For example, FSPTX is the value of the Fidelity Select Technology Portfolio.)

4.5 Results from the More Difficult Problem

Does NM improve GP's ability to solve this problem from the financial analysis domain? Our experimental hypothesis is that NM hybrid will have a higher success ratio then the non-hybrid GP.

Again we used a control group of a set of GP runs without NM. This time the control set had a population size of 561 which is 20 individuals larger than the population of the sets with NM. This allows all sets to perform the same number of calls to the fitness function. Note that in this case we have applied NM to 20 individuals chosen from the top 100 individuals in the population (sorted by fitness scores). The results are shown in Figure 5.

Using the same statistical test as before (Section 4.1) we found that the difference between the non-hybrid set of runs and the set with NM with a THP of 0.000001 was statistically significant. So in this case we can reject the null hypothesis and conclude that NM with this THP has a superior success ratio to that of the non-hybrid GP.

The success ratios of all of the other sets of NM hybrid GP did not have statistically significant differences from the success ratio of the non-hybrid control group, and thus we could not reject the null hypothesis.

We conclude that we can reject the null hypothesis for one set of runs with NM, but still feel that in the future we should do more experiments with larger populations with more runs and allow the runs to continue for more generations.

5 Interpreting the Results

We have demonstrated that numeric mutation can provide an improvement to the GP algorithm as it is applied to these problems. The next step is to understand from whence this benefit derives.

It is apparent that the numeric mutation technique provides a much greater diversity of numeric constants to the GP. Plain GP (without numeric mutation) starts (in generation 0) with a fixed number of numeric constant leaf nodes in the entire population (i.e., in the genospecies). Whenever the selection process causes all copies of a numeric constant to be removed from the population, that numeric constant is effectively lost for the remainder of the run. Thus, with each generation the number of unique numeric constant leaf nodes can never increase and, indeed, typically decreases monotonically. Numeric mutation replaces all of the numeric leaf nodes with new numeric constants in all of the elements to which it is applied. Thus the GP process gains many new numeric constants each generation by using numeric mutation.

We conducted experiments to determine if the steady influx of new numeric constants, alone, accounted for the benefit of the NM technique. We repeated the original experiment, but instead of using NM we completed 1000 GP runs in which 40 elements were selected after each generation in the same way as in numeric mutation, but all of their numeric constant leaf nodes were replaced with new numeric constants. These new constants were selected randomly from the interval $(-1000.0, 1000.0)$ using a uniform distribution. We call this process *numeric replacement*. Numeric replacement is similar to the technique referred to as *small-mutation* in (Harris and Smith 1997) except that numeric replacement concerns only numeric constant leaf nodes, while small-mutation can affect any type of node.

The result of the numeric replacement experiment was that only 278 of the runs were successful by the 50th generation as compared to 328 successful runs with plain GP. To determine if this difference was statistically significant we again used the Large-Sample Statistical Test for Comparing Two Binomial Proportions described above. We determined, with 95% confidence, that numeric replacement produces a statistically significant *degradation* of performance when compared to plain GP. Therefore it is highly probable that the benefit derived from numeric mutation does not derive solely from the influx of new numeric constants, but also from the values of those constants.

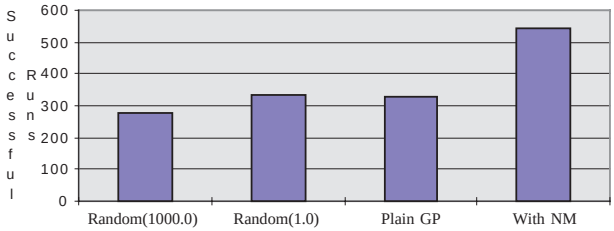

Figure 6 Number of successful runs (of 1000), handling numeric constants four different ways.

A potential criticism of this experiment is that the range (e.g. $(-1000, 1000)$) from which the new constants were chosen in numeric replacement did not correlate well with the problem domain. The function used to generate the regression's target points (see Equation 1) contains no numeric coefficients with an absolute value greater than 1.0. So the range may be introducing numeric constants into the genospecies that are unlikely to be useful in solving the specific symbolic regression problem under study.

To investigate this possibility we again conducted the numeric replacement experiment, but used the range $(-1.0, 1.0)$ from which to select the new constants. Of the 1000 runs, 336 were successful by the 50th generation. This, at least, was more than the 328 successful runs that occurred with the plain GP, but the Large-Sample Statistical Test for Comparing Two Binomial Proportions determined that this improvement is not statistically significant at the 95% confidence level. We again conclude that the benefit of numeric mutation does not derive solely from the influx of a large number of new numeric constants. A summary of all these results is shown in Figure 6. The entries in the figure labeled "Random" correspond to numeric replacement.

Having eliminated other possible explanations, we speculate that the benefit of numeric mutation derives not simply from the introduction of new numeric constants into the genospecies, but also from these new constants being introduced only into s-expressions at locations in genomes where arithmetically similar numeric constants have already demonstrated some measure of success, insofar as they appear in individuals in the top part of the population, as scored by the fitness function. We further speculate that the choice of new numeric constants is further enhanced by making them increasingly more similar to the existing "successful" constants as the population comes closer to finding an acceptable solution.

6 Conclusion and Future Work

We conclude that the use of numeric mutation should be considered for GP problem in which floating point numeric constants are used as terminal nodes. Numeric mutation is easy to implement and does not add significant additional overhead to the GP algorithm.

Several additional experiments are suggested by this work. We plan to see if additional benefit can be derived by applying numeric mutation only to a portion of the numeric constants in selected individuals, and to experiment with alternative methods for determining the temperature factor, such as using the raw score of the individual to be mutated rather than the raw score of the best element in the generation.

References

Evett, M. and T. Fernandez (1997). A distributed system for genetic programming that dynamically allocates processors. Technical Report TR-CSE-97-39. Dept. Computer Science and Engineering, Florida Atlantic University. Boca Raton, FL.

Fernandez, T. (1997). The evolution of numeric constants in genetic programming. Master's thesis. Florida Atlantic University. Boca Raton, FL.

Fernandez, T. and M. Evett (1997*a*). The impact of training period size on the evolution of financial trading systems. Technical Report TR-CSE-97-41. Florida Atlantic University. Boca Raton, FL.

Fernandez, T. and M. Evett (1997*b*). The impact of training period size on the evolution of financial trading systems. In: *GP-97, Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, Eds.). Morgan Kaufmann.

Fernandez, T. and M. Evett (1998). Numeric mutation as an improvement to symbolic regression in genetic programming. In: *Proceedings of Evolutionary Programming '98*. Morgan Kaufman. to appear.

Fogel, D. (1997). The burden of proof. Invited lecture at Genetic Programming 1997, Palo Alto, CA.

Harris, K. and P. Smith (1997). Exploring alternative operators and search strategies in genetic programming. In: *GP-97, Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, Eds.). Morgan Kaufmann. San Francisco, CA, USA. pp. 147–155.

Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi (1983). Optimization by simulated annealing. *Science* **220**, 671–680.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press.

Koza, J. (1997). Tutorial on advanced genetic programming, at genetic programming 1997.

Mendenhall, W. and O. Lyman (1972). *Understanding Statistics*. Duxbury Press. Belmont, CA.

Rumelhart, D.E. and J.L. McClelland (1987). *Parallel Distributed Processing*. Vol. 1. MIT Press. Cambridge, MA.