Final Project - Genetic Programming: Sensitivity to Multiple Parameters

Sarah Bailey

MATH 3220-01

Professor Aleshunas

December 15, 2010

Executive Summary

The purpose of this experiment is to run a series of experiments that can be used to analyze the impact that the parameters of population size, the method of initializing the population, and the breeding operators' probability, have on the convergence of a genetic programming system. Genetic Programming is an evolutionary method of solving problems using natural selections and computers. The goal is to get the computers to solve high ended questions without having to tell the computer how to reach the optimal solution. GP represents the possible solution candidates as a tree that contains nodes of functions and predetermined programs that the computer is already aware of. There is a constant loop in the execution that follows the steps: an initial population is set, a new generation is created by crossover and mutation operations, the fitness of each individual is assigned, and then finally the selection process chooses which individuals with proceed for the future generation. The essential goal of GP is to take a population of candidate solutions and evolve it over multiple selections and random operations to reach an optimal solution that portrays enough randomness to discover new solutions, in a timely manner (Dolan).

GP is an input based process that has multiple parameters that can be altered to adjust how the optimal solution is achieved. In these experiments I work with adjusting the population size parameters, method of initializing the population, and the breeding operators' probability, and analyzing their importance via the results. The genetic programming system that will be used in this experiment is *lil-gp*, a C-language program for developing GP applications. The changes for the three parameters will be applied to the two problems of *Artificial Ant* and *Regression*. *Ant* is a problem designed to move an ant on a grid and finding the most optimal path for finding the most food. *Regression* is a problem that is based on mathematical functions that try to find the most ideal solution in comparison to a desired curve (Zongker, Douglas, Punch, Dr. Bill, and Rand, Bill 10-11). A series of executable runs were done with the adjustment of the population (sizes: 200, 400, 800), the initial population method (full, grow, half and half), and the breeding operators' probability (70% Crossover- 25% Mutation, 80% Crossover-15% Mutation, and 90% Crossover-5% Mutation). These were done to find the impact that these parameters have on the convergence of a genetic programming system with the given algorithms.

After running the multiple experiments, I found that in the *Ant* problem the initial method was the most sensitive, then the population size, and lastly the breeding operator was the least sensitive parameter. For the most part in *Ant*, the convergence of the different parameters remained at a steady increase into its most fit individual and these scores of their fitness level were estimated at an average of .27 on a scale of 0-1. For the *Regression* problem, it was difficult to evaluate which parameter was the most sensitive to this problem; almost all of the curves were relatively close throughout the entire set of generations. The convergence for the *Regression* problem was much faster than that of the *Ant* problem, and its overall average for the fitness score was estimated at .9 on a scale of 0-1.

Problem Description

Genetic Programming is an evolutionary method, based on biological evolution, which is designed to find programs that can solve optimal instruction set problems. In more ordinary terms, it's a technique for computers to automatically solve problems, without having to know or define the structure of the desired solution beforehand. GP represents a population of candidate solutions as trees structures that are composed of elements from a user determined function set and terminal set. The operations that are essential for evolving the new population solutions are crossover (child nodes are exchanged between parent nodes) and mutation (randomly selected sub-tree nodes are modified) (Dolan).

GP contains a set of parameters that can be adjusted by the user to help modify and reach the optimal solution in a timely and correct manner. There are important parameters that GP deals with such as fitness evaluation, selection techniques, breeding operators' probability, tree sizes, population size, and initialization of the population methods. A critical issue arises with the question of what are the appropriate choices of parameters to ensure randomness and diversity, but that can be done in an appropriate time without increasing computational costs of evaluating the fitness of the given population (Banzhaf, Nordin, Keller, and Francone 109).

The purpose of this experiment is to analyze the different effects that the parameters tested (population size, initialization method, and breeding operators' probability) have on the convergence of a genetic programming system. The genetic programming system that will be used in this experiment is *lil-gp*, a C-language program for developing GP applications. *Lil-gp* is a system that has flexible format of parameters, has several selection methods, three genetic operators, optional node and/or depth limitation on the tree size, allows ephemeral random constants, and creates output files that are easily accessible in programs like Microsoft Excel (Zongker, Douglas, Punch, Dr. Bill, and Rand, Bill 6). Since GP's are represented as tree structures, this system will display its findings as a tree which can be interpreted as a program.

A series of experiments will be done with the adjustment of the parameters to both *Artificial Ant* and *Regression* problems. *Artificial Ant* is a problem that represents a grid provided with a trail of food pellets distributed all over. There are two different examples in Ant, but the one I used for these experiments is the Santa Fe Ant trail (32 x 32 grid with 89 food pellets distributed). The population is kept at a constant number and there are a predetermined number of steps allowed to run. Once the population has moved, a set number of cycles, a new generation is born. The ants in the problem are allowed to move left, right, and/or move forward, and there are two finite states in the function set, if there is or is not food ahead. The GP will create a path and measure the ant's fitness by the number of food pellets the ant came into contact with. The other problem that will be examined is *Regression*; this problem is programmed to generate a function which will be comparable to a desired curve. Its fitness is determined by analyzing distance between the y coordinate of a predetermined point on the curve (x, y) and the value for y that is returned by the program. The function set contains: multiply, protected-divide, addition, subtraction, sin, cos, exponential, and protected-log. The terminal set has either one or two members: the input value x, and (optionally) the ephemeral random constants (Zongker, Douglas, Punch, Dr. Bill, and Rand, Bill 10- 11).

Analysis Technique

Genetic Programming, noted as GP, is an automated evolutionary algorithm based technique of creating a computer program that will solve a high level problem statement of a problem ("Evolutionary Computation"). The purpose of GP is to get a computer to perform a task without going through the steps of creating a program for the computer, but rather the computer uses other known programs, depicted in the form of trees, to create a new program that will reach the desired solution. The goal of GP is automatic programming, achieved by taking the given population of computer programs and genetically breeding them by using Darwin's natural selection operations, mutation, crossover, reproduction, and gene duplication and deletion ("What Is Genetic Programming?"). For this experiment *lil-gp* will be the genetic programming system for executing and testing GP; Lil-*gp* is a C-language program that will be used for evolving variable length tree structured GP. The genetic programming system will test the two problems *Artificial Ant* and *Regression*.

The executable process of GP begins with randomly generating an initial population of functions and terminals of the problem. Next, each program of the population is executed and assigned a fitness score

for which they are effective for the problem. After the fitness is scored and assigned, the GP creates a new population by copying the best program, with the highest fitness score, and using the main operations of mutation, and crossover. Once the termination criterion has been reached, the single best program of the population produced is designated the solution program for solving the given high level statement problem ("What Is Genetic Programming?").

The parameters that are important to running *lil-gp* for the *Ant* and *Regression* problems are: limits on the tree size, the selection type, population size, generating the initial population method, and the breeding operators' probability. The parameters that were not tested in this experiment were tree size and the selection type. Since trees can tend to grow very largely, which may cause the evaluation time for each tree to be long, the user is allowed to put limits on the number of nodes and/or adjust the depth of an individual tree; this is why the limits on the tree size is important to the execution (Zongker, Douglas, Punch, Dr. Bill, and Rand, Bill 12). Another important parameter, selection type, is set by the user by choosing one of its types: fitness proportional, (Greedy) over selection, and/or tournament selection. This parameter is responsible for how quickly the population will converge. For this particular experiment we will only use tournament selection type; the selection type where a number of individuals are randomly selected to "compete" in a tournament in which the individuals' fitness scores are compared therefore the "winner" will be the individual with the best fitness amongst the other competitors (Banzhaf, Nordin, Keller, and Francone 132).

The parameter, population size, is one that was tested in the experiments. Population size controls the amount of individuals that will be used as a starting point for finding the optimal solution. Amongst this predetermined population size, individuals can be chosen to continue into the next generations along with being selected for operations. Population size should be adjusted around two different factors, computational time and ensuring there is a large enough population so that all members can be given a chance to be represented and kept in future generations; therefore not creating a bias towards certain more fit individuals. For this experiment I chose to set the population sizes to 200, 500, and 1000. I chose 200 as a starting point, because it was the lowest amount that could be tested that has sufficient number of members in the population. Anything that was lower than 200 would be unstable and not efficient enough to produce an optimal solution. The next test, setting the population size to 500, was a median in which the population space was large enough to ensure a variety of members, and the computational time will remain low. The last test was setting the population size to 1000. This would of course ensure lots of diversity in the population, but the computational time was much longer than the other tests that were ran.

The next parameter that was tested in my experiment was the method of initializing the population. This parameter has a connection to the first testes parameter, population size. The first step in actually performing the GP run is to initialize the population, which contains three types to be chosen: full, growth, and half & half methods. The full method chooses only functions of a tree until a node is at a predetermined maximum depth; instead of selecting nodes randomly form the function and the terminal set (Banzhaf, Nordin, Keller, and Francone 119). The grow method randomly selects nodes from both functional and terminal sets throughout the entire tree until a branch reaches a terminal node (even if the maximum depth has not been reached). The third choice of methods to choose from is the half and half method; this method is very important to GP populations and it is known to enhance population diversity of the structure. The half and half method is a mixture between the growth and full methods; half of the maximum depth is chosen with the grow method, and the other half is chosen by the full method (120).

The last parameter that was tested in my experiment was the crossover and mutation operations' probability of breeding. Crossover is an operation in which it combines two parent's genetic material by swapping a part of one parent with that of the other parent's part (Banzhaf, Nordin, Keller, and Francone 122). Mutation takes an individual and randomly selects a point in the tree, where it will remove that point and its following nodes, and replace its subtree with a new randomly generated subtree (125). This experiment tested the change in the probability of these two operations occurring in each generational run. The three tests that were done had the breeding probabilities set at: 70% Crossover- 25% Mutation, 80% Crossover- 15% Mutation, and 90% Crossover-5% Mutation; during all these runs the third operator of reproduction was not tested and therefore remained constant at 5%. Amongst these different probabilities that were chosen for the experiments, the crossover operator remained significantly the highest because it is the most common and effective operator for GP.

Assumptions

While examining the graphs, I assumed that a parameter was sensitive if the overall curve of each different experiment was varied amongst the three trials and if the curve was closer together then the parameter was less sensitive (therefore little change in the results).

Results

**Attached is the graphs related to the experiments*

After performing the multiple runs after adjusting the parameters, the results were represented as a graph for each parameter adjustment. For the *Ant* problem I found that the initialization of population method was the most sensitive, then the population size, and the breeding operator probability as least sensitive.

For the initialization of the population method, the grow curve resulted in a higher fitness individual as its optimal solution (about .89 fitness score), but the convergence of this graph was very quickly. The cause of this quick convergence is due to the fact that the grow method doesn't ensure that there is the maximum depth of a tree, therefore the trees can remain rather small and not get enough diversity throughout its trees. The full method curve resulted with an individual with the least fitness score, but the convergence was more evenly distributed (increased fitness level about every 200 generations) throughout the generations and reached its maximum (about .22 fitness score) in the last 100 generations. This curve's shape is caused by the fact that the trees are always kept at a full maximum depth, which is a reason why the curve isn't perfectly smooth, and represents diversity in its generations. Finally the half and half method created a curve that was expected; half way similar to the full method (had a more steady increase in the fitness score) and the grow method (tried to converge rather quickly), while remaining in between the full and grow curves.

The next most sensitive parameter for the *Ant* problem was the alteration in the population size. The curve for population size of 200 had the lowest fitness score compared to the other two, although it did not allow for much diversity in its population. The curve for the population size of 500 showed the best results for the most fit individual, compared to the other two population sizes, and its jump in shape proves enough diversity throughout the population. The curve for the 1000 population size shows a lower fitness score than the 500 size, but the increase/change in the shape occurred more often, which represents even more diversity than the other two population sizes.

The least sensitive parameter for the *Ant* problem was the crossover and mutations operations' probability. This makes sense because the *Ant* problem is based on moving an ant right, left, and/or forwards, which these functions do not have an order in which they must occur. The results concluded

that the breeding percentages of test 1's curve (70% Crossover- 25% Mutation), was very consistent and constant except for a large increase for a short number of generations. The curve for test 2 (80% Crossover- 15% Mutation) showed more diversity, and it was smooth with its transitions to the most fit individual, compared to the other two tests. Finally the curve for the test 3 (90% Crossover- 5% Mutation) showed the an even more smooth graph into its most fit individual, but its overall highest fitness score was less than the previous test.

For the *Regression* problem, I found that the different adjusted parameters were difficult to determine which factor was the most and least sensitive. It is quite difficult to distinguish the difference between the adjust parameters for each parameter, but the overall conclusion is that the convergence occurs quickly, and the fitness score for each experiment is estimated at about .9 out of 1. For the all of the parameters adjusted in *Regression*, the convergence of each curve was much faster and earlier than the *Ant* problem due to the basis of the problem set up; the problem revolves around solving mathematical problems with different functions and set inputs, therefore the time to reach an ideal solution is much faster than that of the ant, which has many more options of possible steps. This conclusion is different then what I had expected especially for the operators probability parameter; I thought that this parameter would be the most sensitive due to the fact that only certain operators can be performed at certain times, and therefore the curve would be more different amongst the changed settings.

Issues

I thought that the most sensitive parameter in the *Regression* problem would be the breeding operators' probability because this problem is based on using functions such as subtraction and division, which are sensitive to the order in which they are performed; this was not the results that I had reached.

Appendices

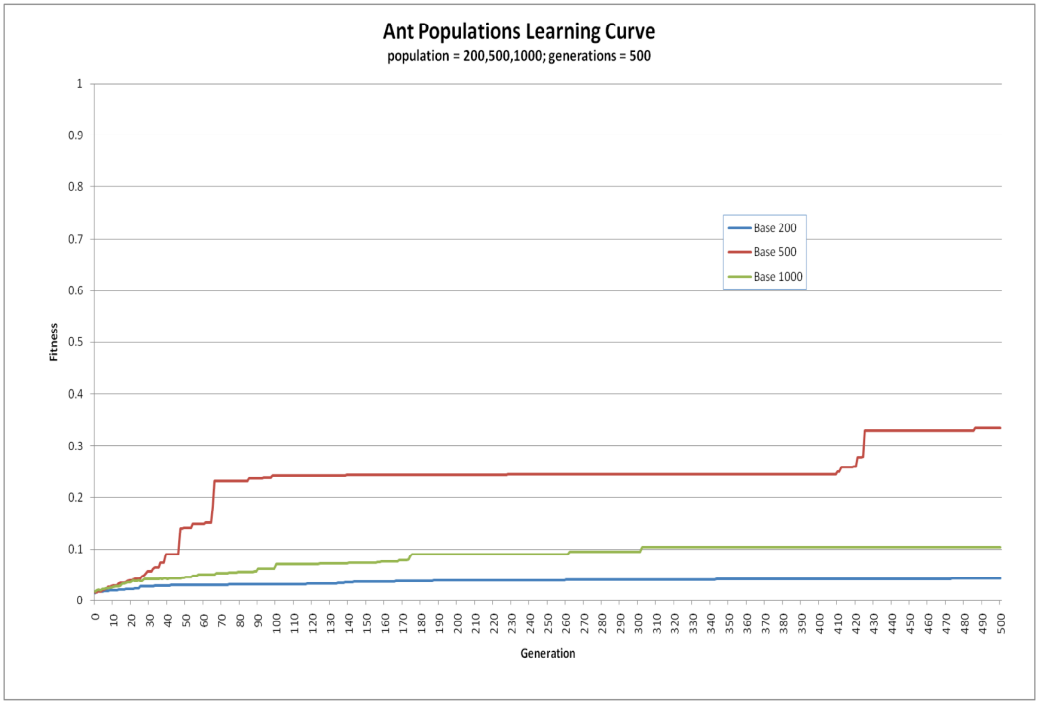
Banzhaf, Wolfgang, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming~ An Introduction*. San Francisco, California: Morgan Kaufmann, 1998. Print.

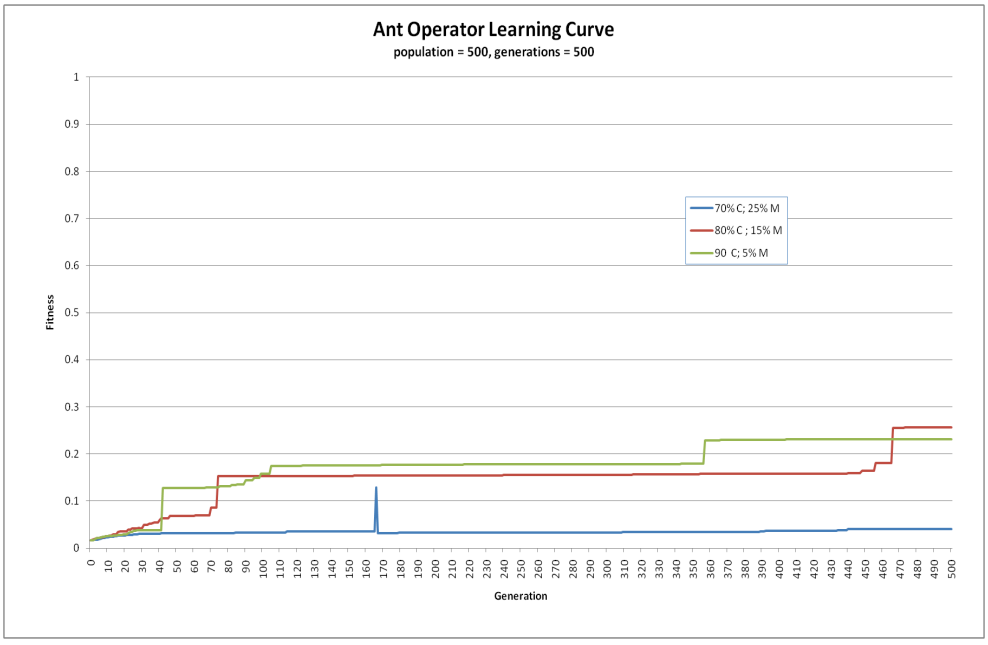
"*Genetic programming*."Dolan, Kevin. 2009.Web. 13 Dec. 2010. <http://geneticprogramming.us/What_is_Genetic_Programming>

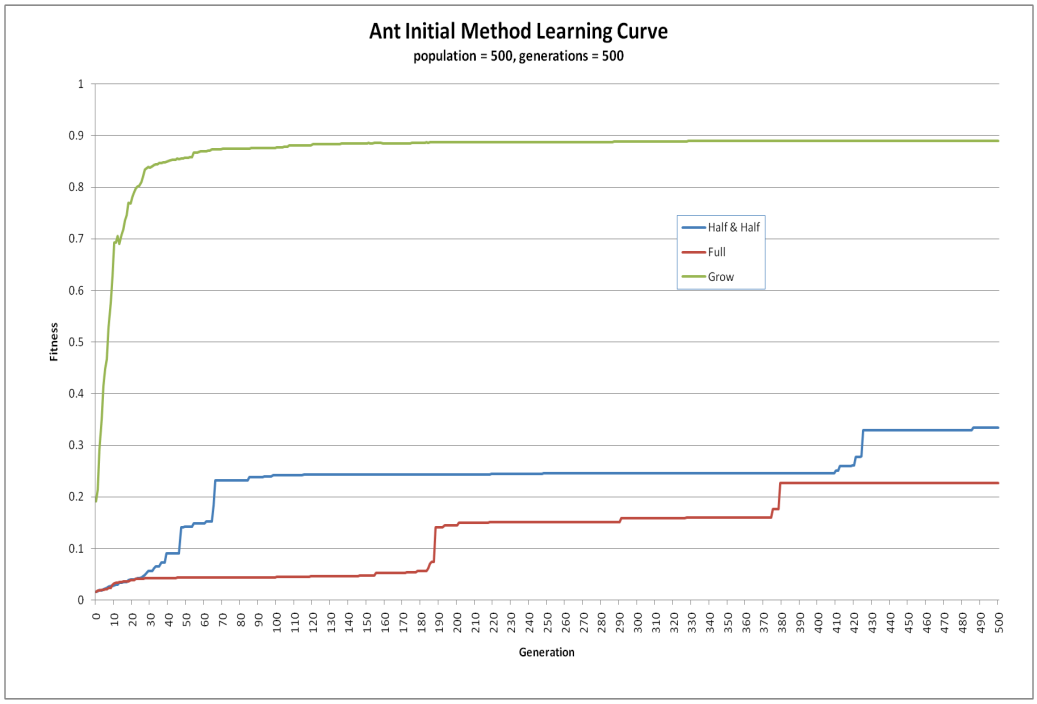
"Evolutionary Computation." *Wikipedia*. 14 Oct. 2010. Web. 13 Dec. 2010. <http://en.wikipedia.org/wiki/Evolutionary_computation>.

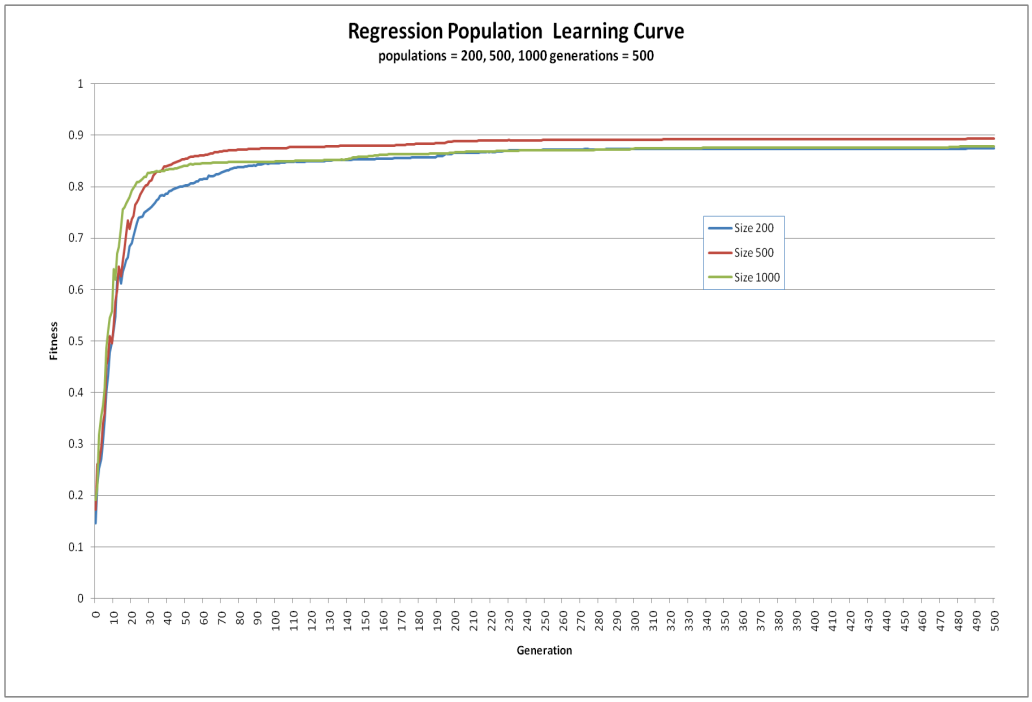
"What Is Genetic Programming?" *Genetic Programming*. 27 Aug. 2003. Web. 13 Dec. 2010. <http://www.genetic-programming.com/gpanimatedtutorial.html>.

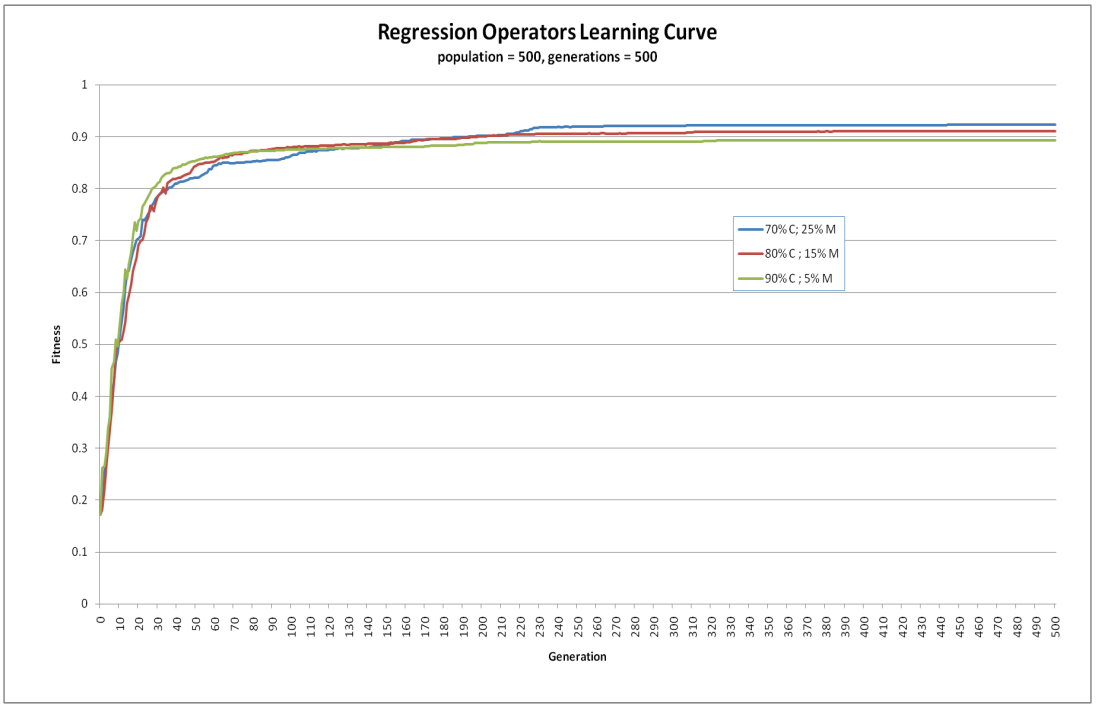
Zongker, Douglas, Punch, Dr. Bill, and Rand, Bill. *lil-gp 1.01 User's Manual.* Michigan State University, 1995. PDF

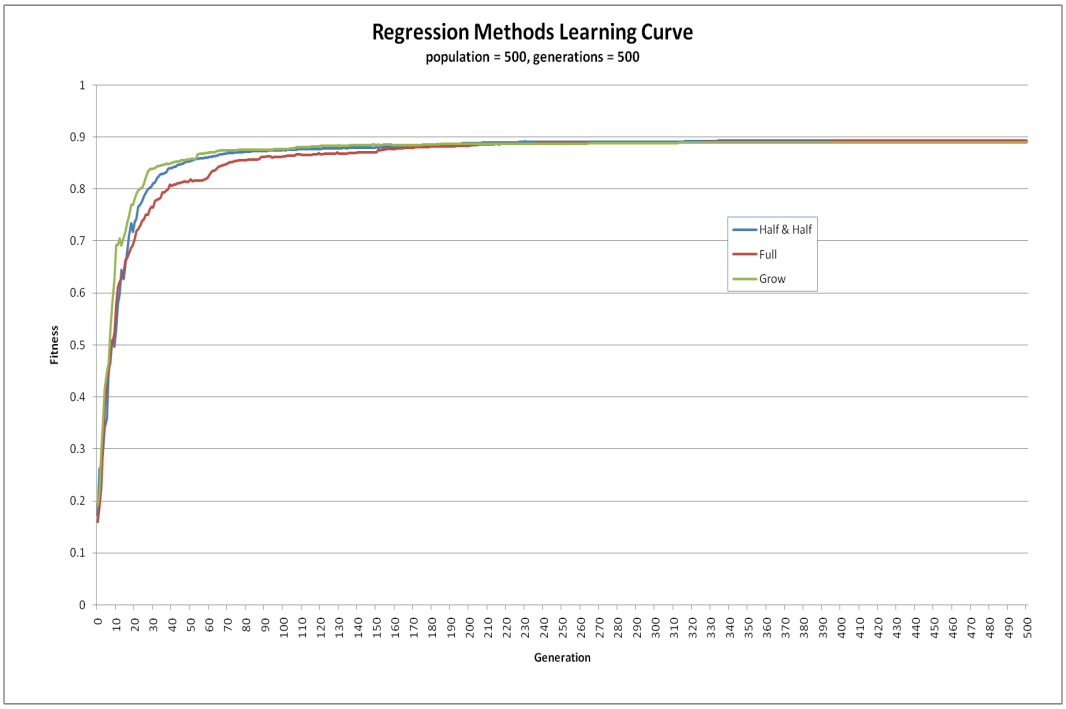

Ant Populations Learning Curve
population = 200,500,1000; generations = 500
Fitness
Generation
Base 200
Base 500
Base 1000


Ant Operator Learning Curve
population = 500, generations = 500
Fitness
Generation
70% C; 25% M
80% C ; 15% M
90 C; 5% M


Ant Initial Method Learning Curve
population = 500, generations = 500
Fitness
Generation
Half & Half
Full
Grow


Regression Population Learning Curve
populations = 200, 500, 1000 generations = 500
Fitness
Generation
Size 200
Size 500
Size 1000


Regression Operators Learning Curve
population = 500, generations = 500
Fitness
Generation
70% C; 25% M
80% C ; 15% M
90% C ; 5% M


Regression Methods Learning Curve
population = 500, generations = 500
Fitness
Generation
Half & Half
Full
Grow