Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

Executive Summary:

The concept of Genetic Programming (GP) is best defined as "it is aspired to induce a population of computer programs that improve automatically as they experience the data on which they are trained. Accordingly, GP is part of the very large body of research called machine learning (ML)." (Banzhaf, Nordin, Keller, & Francone, 1998)

In other words, "GP is an evolutionary algorithm in the field of artificial intelligence. It is inspired by biological evolution in order to find computer programs that perform a user-defined task. The GP is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task." (http://en.wikipedia.org/wiki/Genetic_programming#cite_note-2)

GP determines how well a program is by running it and then compare it to an ideal (usually the original program). The comparison between the program that was run and the ideal program is expressed in a numerical value called "fitness". According to the fitness value, the program that does well is chosen to breed and produce new programs for the new generation. And the primary genetic operators that are used to create new programs from existing ones are called the crossover and the mutation. (Banzhaf, Nordin, Keller, & Francone, 1998)

Now that the definition of Genetic Programming is clear, it is essential to list the important elements shared by most GP systems. These important elements are best defined by (Banzhaf, Nordin, Keller, & Francone, 1998) as:

1. Stochastic decision making: GP uses pseudo-random numbers to mimic the randomness of natural evolution. As a result, GP uses stochastic process and probabilistic decision making at several stages of program development.
2. Program structures: GP assembles variable length program structures from basic units called functions and terminals. Functions perform operations on their inputs, which are either terminals or outputs from other functions. The actual assembly of the programs from functions and terminals occurs at the beginning of a run, when the population is initialized. Figure 2
3. Genetic operators: GP transforms the initial programs in the population using genetic operators. Crossover between two individual programs is one principal genetic operator in GP. Other important operators are mutation and reproduction. Figure 1
4. Simulated evolution of a population by means of fitness-based selection: GP evolves a population of programs in parallel. The driving force of this simulated evolution is some form of fitness-based selection. Fitness-based selection determines which programs are selected for further improvements. "Fitness is the measure used by GP during simulated evolution of how well a program has learned to predict the output(s) from input(s) – that is, the features of the learning domain." ((Banzhaf, Nordin, Keller, & Francone, 1998, p. 126)

Therefore, the purpose of the fitness value is to determine which individuals to be allowed to multiply and evolve and which individuals need to be removed from a specific population. (Banzhaf, Nordin, Keller, & Francone, 1998)

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

The Artificial Ant experiment is chosen to run for GP in this analysis. In this experiment there is a grid provided with a trail of food pellets distributed over the grid. The trail Santa Fe is chosen for this experiment and its dimensions are 32x32 with 89 food pellets. The GP program generates a path by walking through this map. It is allowed to run for some number of time-steps (400). Then, the fitness of the program is measured by the number of food pallets "run over" by the ant. Each terminal costs one time step to evaluate, each function takes no time.

This analysis examined the behavior of populations and their fitness by running the condensed version of all statistics ten times for three parts of the GP experiment; the Artificial Ant. In each part of the experiment, the analyzer chose three different populations (1000, 3000, and 5000) but used 500 generations for all populations.

After running GP for these three population sizes, six output files appear and they are:

.sys → general information about the run

.gen → statistics on tree size and depth

.prg → statistics on fitness and hits

.bst → information about a current best-of-run individual(s)

.his → history of the .bst file

.stt → condensed version of all statistics

The .stt file is very important for the analyzer's attempt to measure the fitness of the program because it has one line per subpopulation per generation, each line consist of 20 apace-separated numbers (columns). The most important column for this experiment is the fourth column of the .stt file, and it represents the standardized fitness of best-of-generation individual.

The Ant experiment runs in three parts: The first part with population size of 1000 the second part with population size of 3000 and the third part with population size of 5000. But the same steps are used to run and analyze the data is each part. So, the data analyzing process is practically is: after running each population size for ten times, take the data from the fourth column (fitness) of the .stt file of each of the ten runs and average them using the excel program. After obtaining the average of the fitness for all the ten runs, graph it where the x-axis represents the generations and the y-axis represents the fitness. The curves on the graph should tell you the truth about the behavior of each population size and their fitness.

PROBLEM DISCRIPTION:

The concept of Genetic Programming (GP) is best defined as "it is aspired to induce a population of computer programs that improve automatically as they experience the data on which they are trained. Accordingly, GP is part of the very large body of research called machine learning (ML)." (Banzhaf, Nordin, Keller, & Francone, 1998)

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

GP is usually used for the machine learning system that develops tree structures. Before the year 1992, many researchers used genetic or evolutionary operators to solve their computer programs, but they didn't survive to this day because they were incompetent. But in 1992, a PH.D graduate from University of Michigan, John Koza, wrote a thesis titled (Genetic Programming. On the Programming of Computers by Means of Natural Selection." And he was the first one to recognize the genetic programming as a new revolutionary discovery, and his statement was based on his research of evolving tree structures. (http://www.genetic-programming.com/johnkoza.html#anchor6007605)

In other words, "GP is an evolutionary algorithm in the field of artificial intelligence. It is inspired by biological evolution in order to find computer programs that perform a user-defined task. The GP is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task." (http://en.wikipedia.org/wiki/Genetic_programming#cite_note-2)

In GP, populations of computer programs are evolved. And just like nature, GP is a random process that doesn't have guaranteed results, but it finds the best solutions for problems by evolving the programs; generation by generation. (http://cswww.essex.ac.uk/staff/rpoli/gp-field-guide/121WhereGPhasDoneWelll23.html)

GP determines how well a program is by running it and then compare it to an ideal (usually the original program). The comparison between the program that was run and the ideal program is expressed in a numerical value called "fitness". According to the fitness value, the program that does well is chosen to breed and produce new programs for the new generation. And the primary genetic operators that are used to create new programs from existing ones are called the crossover and the mutation. (Banzhaf, Nordin, Keller, & Francone, 1998)

Crossover works by changing two, or more, programs by combining them together making a new program, just like the biological making of a child; after the mother's egg is combined with the father's sperm, the child's DNA is already being formed. On the DNA there are locations called the "genes" which usually determine the color of the eye for example. The result of is that the baby may look like the father, the mother, both, or neither. And that is because the DNA of the baby is the result of combining the DNA of the parents, and during the combination of both DNAs, the locations of the "genes" will vary. "The predominant operator used in GP is crossover. In fact, the crossover operator is the basic of the GP building block hypothesis, which is an important part of the basis upon which GP makes its claim to be more than a massively paralleled hill climbing search. In crossover, two parents are chosen and portion from each parent is exchanged to form children. The idea is that useful building blocks for the solution of a problem are accumulated in the population and that crossover permits the aggregation of good building block hypothesis is correct, and then GP search should be more efficient than other machine learning search techniques." (Banzhaf, Nordin, Keller, & Francone, 1998) Figure 1

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

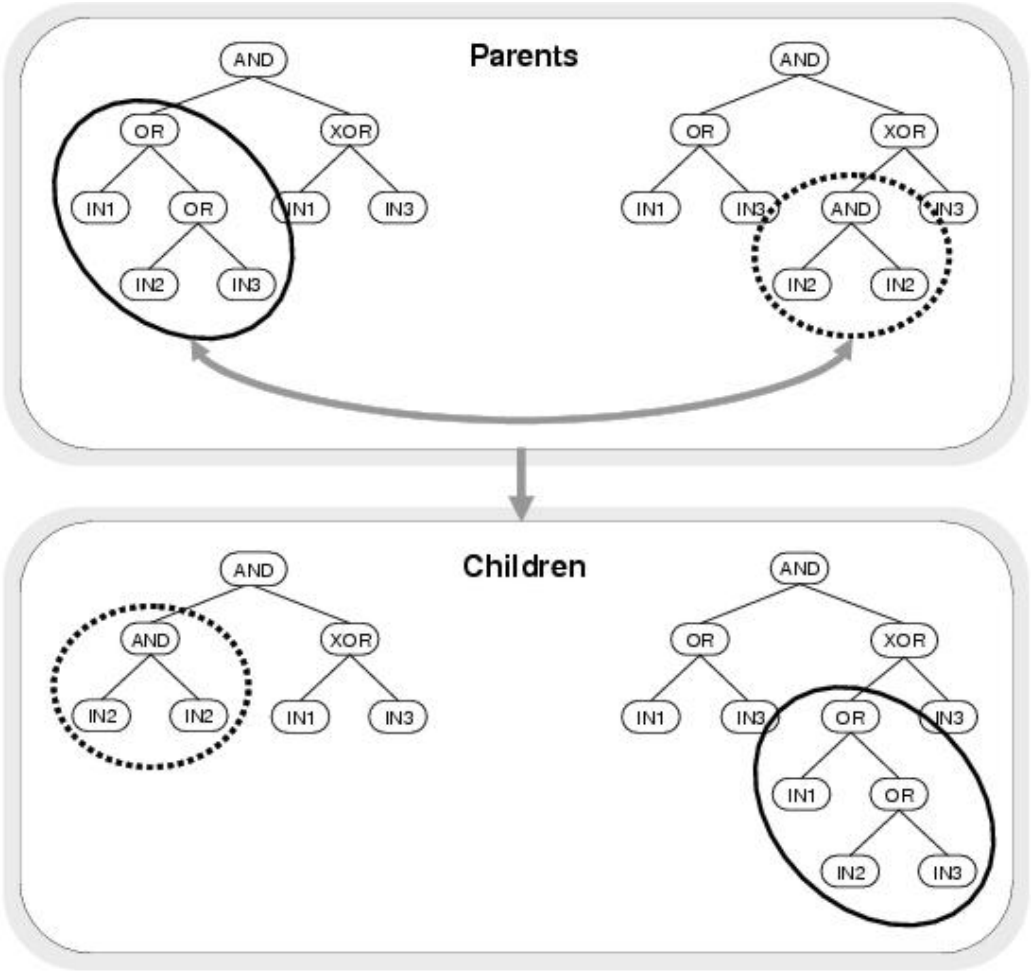

Figure 1: Sub-tree crossover creating new programs from existing programs. Sub-trees are selected randomly from two existing parse trees and are then swapped to produce child parse trees. (http://www-users.york.ac.uk/~mal503/common/thesis/c6.html)

Mutation, on the other hand, is the other main genetic operator. It is not as popular as crossover, but it is very important for GP. Mutation is defined as any random manipulation that can be performed on a single program. (Banzhaf, Nordin, Keller, & Francone, 1998)

Now that the definition of Genetic Programming is clear, it is essential to list the important elements shared by most GP systems. These important elements are best defined by (Banzhaf, Nordin, Keller, & Francone, 1998) as:

1. Stochastic decision making: GP uses pseudo-random numbers to mimic the randomness of natural evolution. As a result, GP uses stochastic process and probabilistic decision making at several stages of program development.

2. Program structures: GP assembles variable length program structures from basic units called functions and terminals. Functions perform operations on their inputs, which are either terminals or outputs from other functions. The actual assembly of the programs from functions and terminals occurs at the beginning of a run, when the population is initialized.

For example:

Figure 2 represent the GP program (2.2 – X / 11 + 7 * COS(Y)) in a syntax tree. The constants and the variables in the program (X, Y, 2.2, 11, and 7*COS(Y)) are the leaves of the tree, but in GP they called terminals, while the arithmetic operations (+, -, * and /) are internal nodes called functions. (http://en.wikipedia.org/wiki/Genetic_programming#cite_note-2)

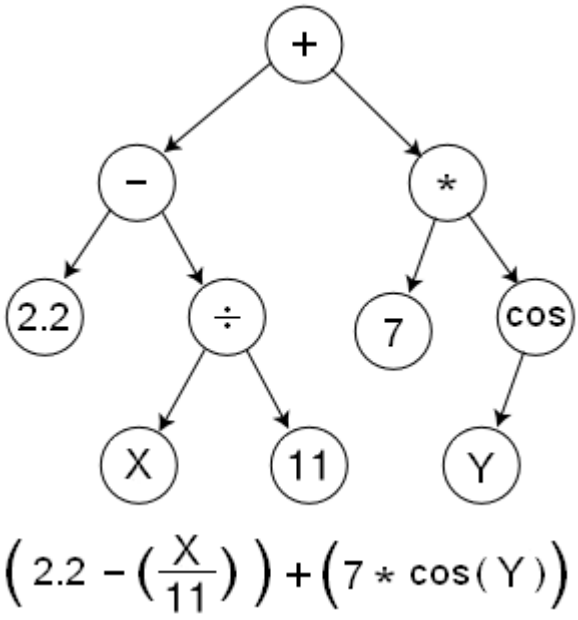

Figure 2: GP program (2.2 – X / 11 + 7 * COS(Y)) represented in a syntax tree.

3. Genetic operators: GP transforms the initial programs in the population using genetic operators. Crossover between two individual programs is one principal genetic operator in GP. Other important operators are mutation and reproduction. Figure 1
4. Simulated evolution of a population by means of fitness-based selection: GP evolves a population of programs in parallel. The driving force of this simulated evolution is some form of fitness-based selection. Fitness-based selection determines which programs are selected for further improvements. “Fitness is the measure used by GP during simulated evolution of how well a program has learned to predict the output(s) from input(s) – that is, the features of the learning domain.” ((Banzhaf, Nordin, Keller, & Francone, 1998, p. 126)

Therefore, the purpose of the fitness value is to determine which individuals to be allowed to multiply and evolve and which individuals need to be removed from a specific population. (Banzhaf, Nordin, Keller, & Francone, 1998)

ANALYSYS TECHNIQUE:

This analysis examined the behavior of populations and their fitness by running the condensed version of all statistics ten times for three parts of a GP experiment; the Artificial Ant. In each part of the experiment, the analyzer chose three different populations (1000, 3000, and 5000) but used 500 generations for all populations.

Artificial Ant experiment:

In this experiment there is a grid provided with a trail of food pellets distributed over the grid. The trail Santa Fe is chosen for this experiment and its dimensions are 32x32 with 89 food pellets. The GP program generates a path by walking through this map. It is allowed to run for some number of time-steps (400). Then, the fitness of the program is measured by the number of food pallets "run over" by the ant. Each terminal costs one time step to evaluate, each function takes no time. The function set has four members. The first is (if-food-ahead), which has two arguments; one to be performed if there is food in front of the ant, the other otherwise that is written in the form (if-food-ahead (arg1-true-subtree) (arg2-false-subtree)). The other three functions are prog2 (2 args), prog3 (3 args), and prog4 (4 args). Each of these functions simply executes its children from left to right. The terminal set has 3 members: move, which moves the ant one step forward. Left, which turns the ant left, and right, which turns the ant right. (Zongker, Punch, & Rand, 1996)

According to (Banzhaf, Nordin, Keller, & Francone, 1998) the problem has three parameters:

<App.maxtime> it is an integer that represent the maximum number of time steps before the ant "times out."

<app.trail> it is the name of a file depicting the "trial" of food for the run. the first line of the first line of the file should have the x and y dimensions. Remaining lines should have a "#" to indicate a square with food, any other character to a blank square.

<app.use_prog4> it is a binary type file. It determines whether or not the PROGN4 should be included in the function set.

The Santa Fe Trail is provided in the ant/directory (santafe.trl). in addition, the .bst file contains the number of time steps used by the ant, the food collected, and a picture of the trail. Blank squares are shown by a ".", uneaten food is a "#", squares the ant passed through have an "x", and the final location and facing direction of the ant are indicated by a "N", "S", "E", or "W" in the appropriate square.

Running the GP program provides a number of output files that contains statistics on tree size fitness for each generation. The filenames are produced by appending a three-character extension to the value of the output.basename parameter. The filenames are:

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

.sys → general information about the run

.gen → statistics on tree size and depth

.prg → statistics on fitness and hits

.bst → information about a current best-of-run individual(s)

.his → history of the .bst file

.stt → condensed version of all statistics

The .stt file is very important for the analyzer's attempt to measure the fitness of the program because it has one line per subpopulation per generation, each line consist of 20 apace-separated numbers (columns). The most important column for this experiment is the fourth column of the .stt file, and it represents the standardized fitness of best-of-generation individual.

We start the experiment by running each part of the experiment only once doesn't. The first part with population size of 1000 the second part with population size of 3000 and the third time with population size of 5000. But the same steps are used to run and analyze the data is each part. Take the data from the fourth column (fitness) of the .stt file of run and average them using the excel program. After obtaining the average of the fitness for the run, graph it where the x-axis represents the generations and the y-axis represents the fitness, the graph will look similar to the

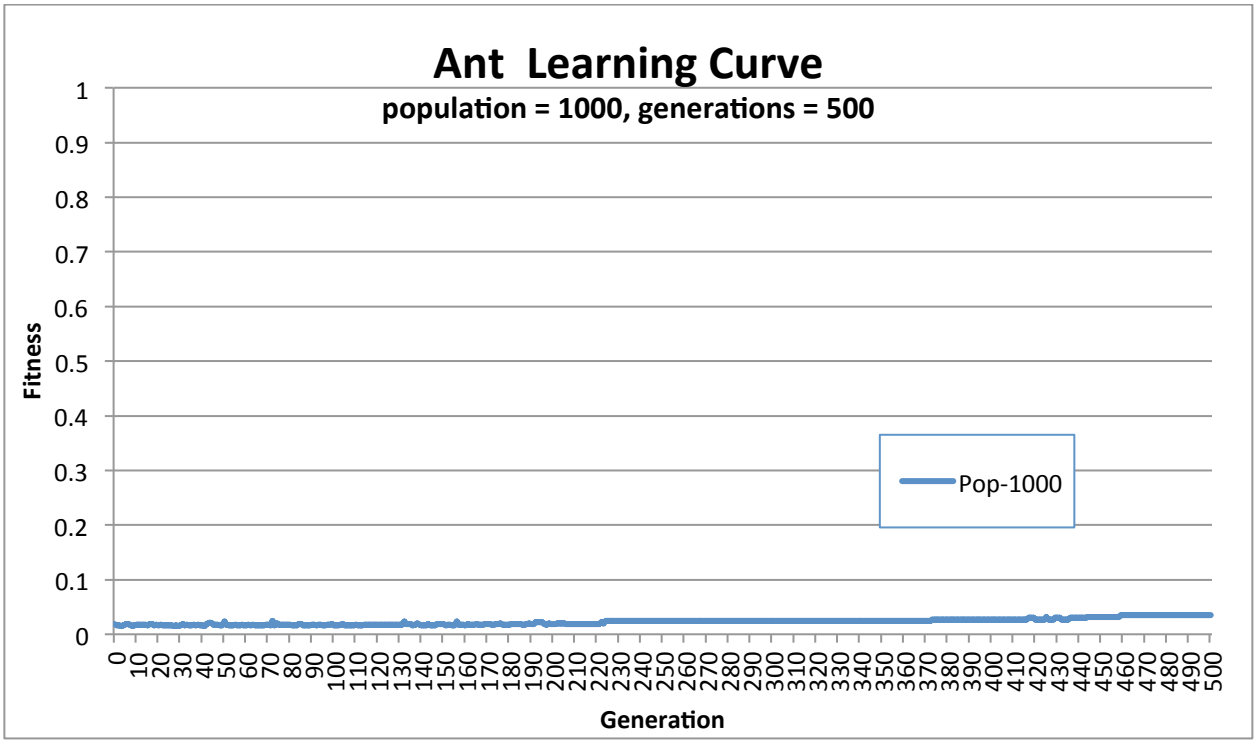

Figure 3: Artificial Ant experiment, part 1 (run once), and population size 1000

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

This graph shows that the fitness of the generations of population 1000 is about consistent, there are few peaks on the curve that resemble increase and decrease of the fitness as the generations increase, but these peaks as very small to the point that they could be consider negligible.

After running the second part of the experiment with population size 3000, calculating the average of its run. Graph the average from the second run with the average from the first run; you will get a graph looks like

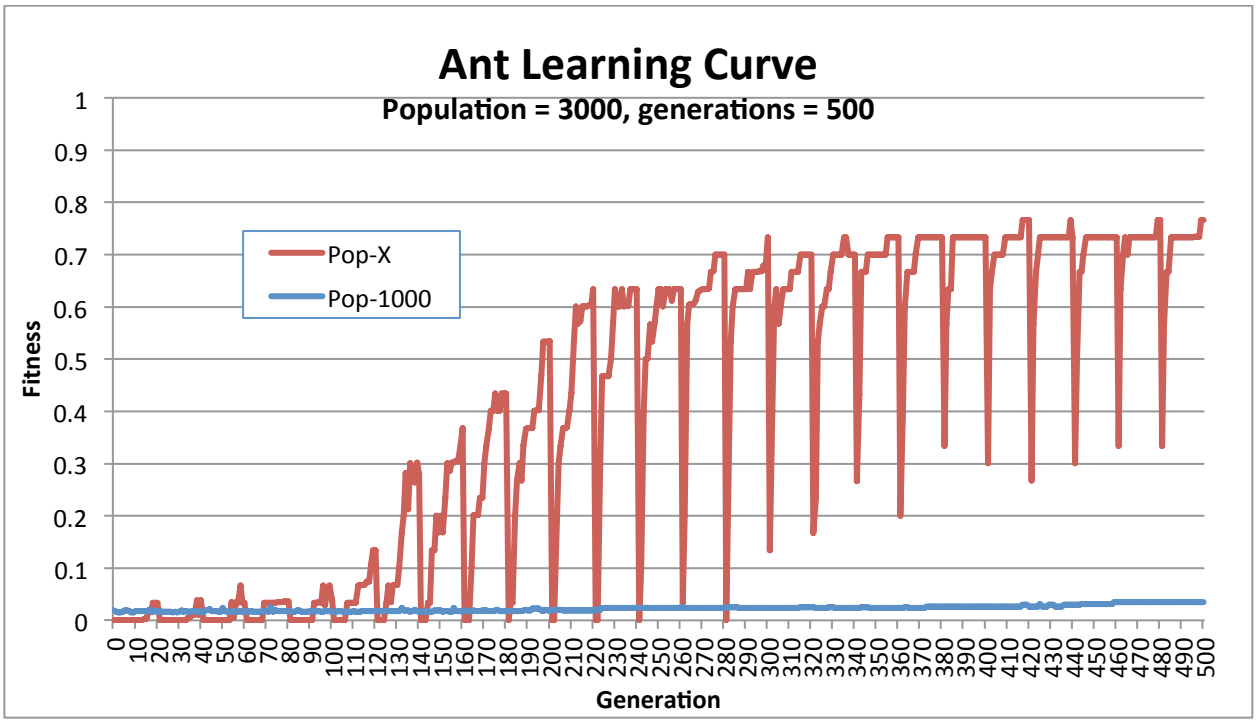

Figure 4: Artificial Ant experiment, part 2 (run once), population size 1000 and 3000

This graph shows dramatic peaks and jumps of the fitness of population size 3000, which makes it harder to determine if the fitness of this particular population is good or bad.

The last part of the experiment uses a population size of 5000. In this part of the experiment, graph the average of the run with the averages from part 1 and 2 and the curve is going to look similar to Figure 5:

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

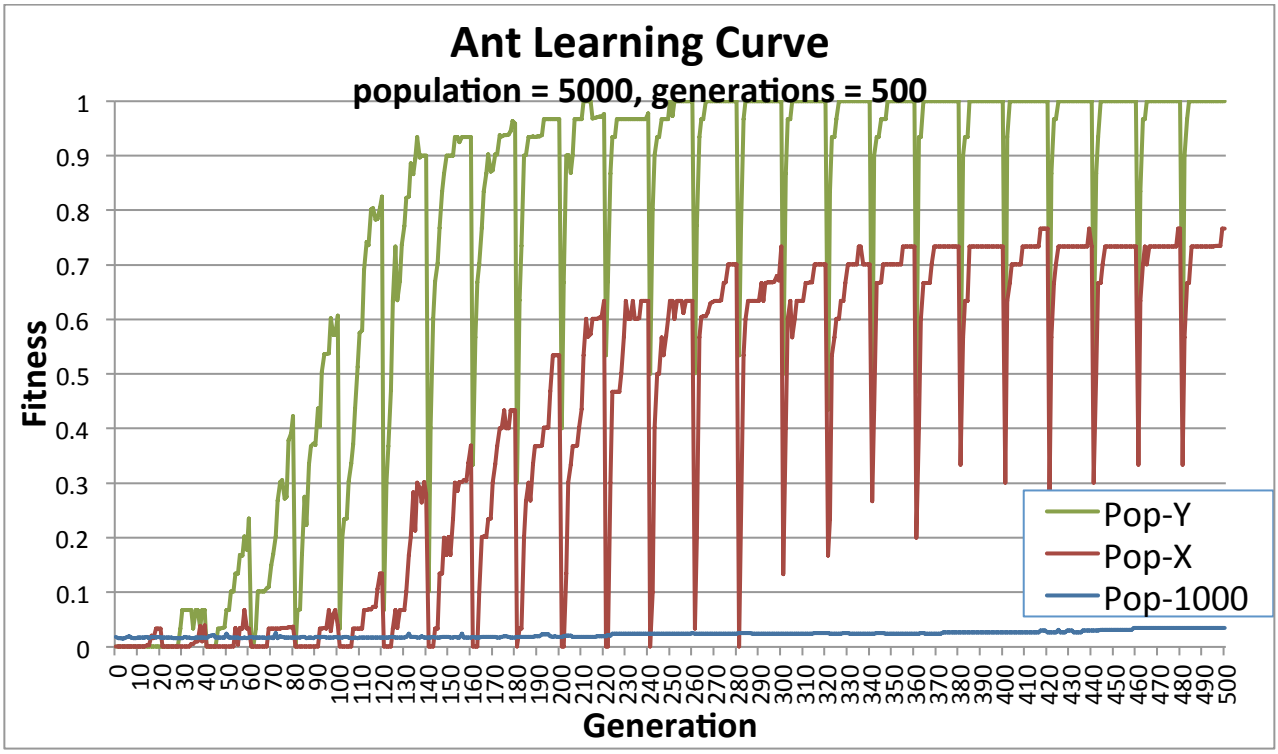

Figure 5: Artificial Ant experiment, part 3 (run once), and population size 1000, 3000, and 5000

This graph shows even more dramatic peaks and jumps in the fitness curve of the population size 5000, which makes it very difficult to determine if the fitness of this particular population is good or bad.

Looking back at the graph results in the last three runs of the Artificial Ant experiment, it is obvious that an answer can't be made to determine the fitness of a specific population size. More data is needed in order to a get a more accurate result. The solution is in repeating the experiment and this time each part is run ten times.

The Artificial Ant experiment with ten runs in each part of the three:

The Ant experiment runs in three parts: The first part with population size of 1000 the second part with population size of 3000 and the third time with population size of 5000. But the same steps are used to run and analyze the data is each part. So, the data analyzing process is practically is: after running each population size for ten times, take the data from the fourth column (fitness) of the .stt file of each of the ten runs and average them using the excel program. After obtaining the average of the fitness for all the ten runs, graph it where the x-axis represents the generations and the y-axis represents the fitness, just like Figure 6 below:

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

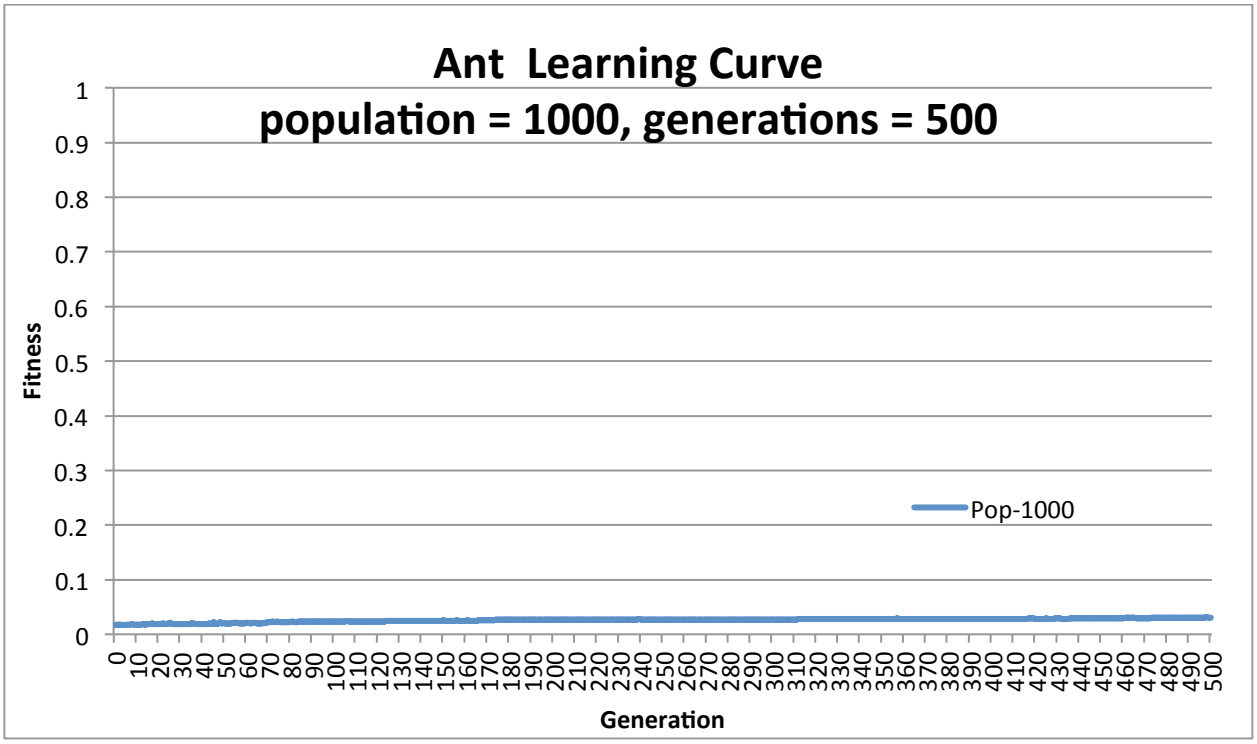

Figure 6: Artificial Ant experiment, part 1-population size 1000

This graph shows that the fitness of the generations of population 1000 is about consistent, there are few peaks on the curve that resemble increase and decrease of the fitness as the generations increase, but these peaks as very small to the point that they could be consider negligible.

After running the second part of the experiment with population size 3000, calculating the average of its ten runs. Graph the average from the second run with the average from the first run, you will a graph just like Figure 7 below:

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

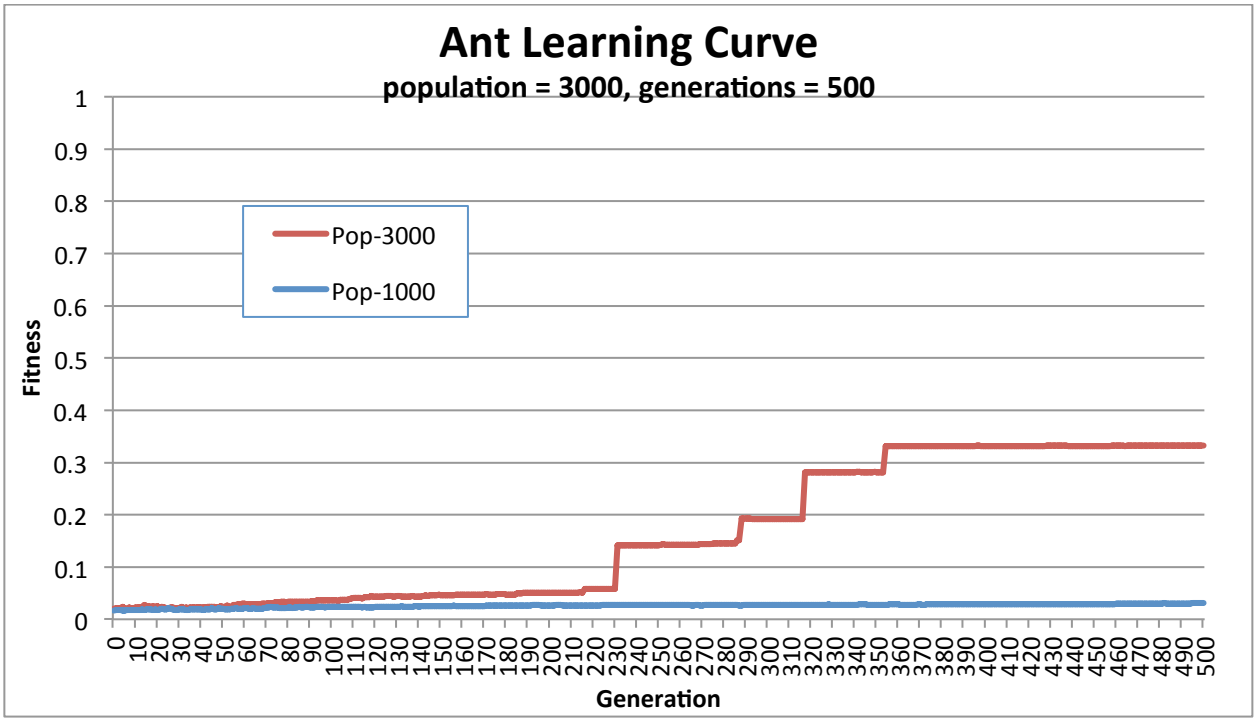

Figure 7: Artificial Ant experiment, part II- with population size 1000 and 3000

This graph shows that the fitness of the generations of population 1000 is about consistent, there are few peaks on the curve that resemble increase and decrease of the fitness as the generations increase, but these peaks as very small to the point that they could be consider negligible. This can't be said for the fitness of generations of population 3000, the fitness increases as the generations increase and it obvious from the jumps of the curve. Also, notice that the fitness is about consistent up until generation 240 where the fitness increases a great deal, and then it goes back to being consistent after it passes generation370.

The last part of the experiment uses a population size of 5000. In this part of the experiment graph the average of the ten runs with the averages from part 1 and 2 and the curve is going to look just like the graph in Figure 8:

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

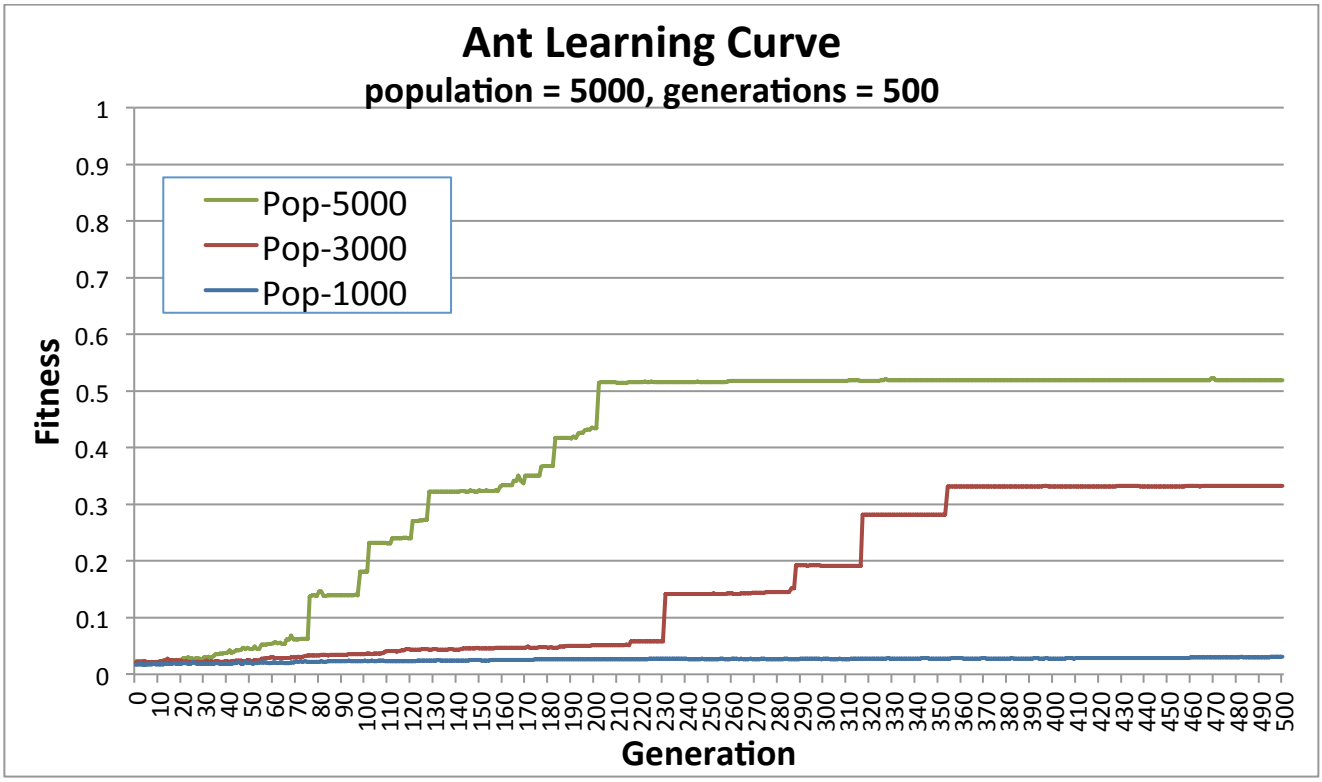

Figure 8: Artificial Ant experiment, part III- population size 1000, 3000, and 5000

In the graph above, the fitness of generations of population 5000 increases rapidly at the beginning of the 500 generations, but right when it reaches generation 200 it becomes very consistent.

The graphs are very important for this experiment, because they explain the behavior and relationship among generation size, population size and their fitness.

RESULTS:
Based on the analysis technique above, these results found are:

1. When the experiment is run once:
   a. the fitness of the generations of population 1000 is about consistent, there are few peaks on the curve that resemble increase and decrease of the fitness as the generations increase, but these peaks as very small to the point that they could be consider negligible. Figure 3
   b. Peaks and jumps of the fitness of population size 3000, which makes it harder to determine if the fitness of this particular population is good or bad. Figure 4
   c. Even more dramatic peaks and jumps in the fitness curve of the population size 5000, which makes it very difficult to determine if the fitness of this particular population is good or bad. Figure 5
2. When the experiment is run ten times:

a. the fitness of the generations of population 1000 is about consistent, there are few peaks on the curve that resemble increase and decrease of the fitness as the generations increase, but these peaks as very small to the point that they could be consider negligible.
b. As the fitness of the generations of population 1000 is about consistent, this can't be said for the fitness of generations of population 3000, the fitness increases as the generations increase and it obvious from the jumps of the curve. Also, notice that the fitness is about consistent up until generation 240 where the fitness increases a great deal, and then it goes back to being consistent after it passes generation 370.
c. The fitness of generations of population 5000 increases rapidly at the beginning of the 500 generations, but right when it reaches generation 200 it becomes very consistent.

3. Using a bigger population and more runs will eventually smooth out the sharp peaks and jumps in the fitness curve on the graph of all three parts of the experiment.
4. Large populations run a lot slower in GP.

ASSUMPTIONS:

The analysis was conducted using the following assumptions:

- The training set is the .stt output file.
- The original data is unchangeable. Meaning the fitness data in the .stt output file is fixed from start to end.
- There are no missing data.
- GP is very accurate to use while analyzing the Artificial Ant experiment.

ISSUES:

While analyzing the Artificial Ant experiment, two main issues came up: 1) it was hard to decide which size population will give better fitness values and a clearer curve of each part of the experiment. And 2) the difficulty to determined how many runs needed to be made for the experiment.

Additional, a minor issue that appeared while running the data was related to the time consumption for each of the different populations that was run in GP; Large populations takes more time to run.

APPENDECIES:

(Sastry, O'reilly, Goldberg, & Hill, 2003)

(Michalewicz, 1996)

Bibliography

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic Programming, An Introduction On the Automatic Evolution of Computer Programs and Its Applications.* Kaufmann Publishers, Inc.

Sara Sabaa
Final project
Artificial Ant experiment
Due 12/14/2011

http://cswww.essex.ac.uk/staff/rpoli/gp-field-guide/121WhereGPhasDoneWelll23.html. (n.d.).

http://en.wikipedia.org/wiki/Genetic_programming#cite_note-2. (n.d.).

http://www.genetic-programming.com/johnkoza.html#anchor6007605. (n.d.).

http://www-users.york.ac.uk/~mal503/common/thesis/c6.html. (n.d.).

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.*

Sastry, K., O'reilly, U.-M., Goldberg, D. E., & Hill, D. (2003). *Building-Block supply in Genetic Programming.* Urbana, IL.

Zongker, D., Punch, D. B., & Rand, B. (1996, March 27). lil-gp 1.01 User's Manual .