GENETIC PROGRAMMING

The Artificial Ant Experiment

By: Sara Sabaa

Outline:

- Definitions of GP.
- Elements of GP
- GP operators
- Examples
- Artificial Ant experiment
- Graphs
- Results
- References

What is “Genetic Programming”?

- 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).”

What is Genetic Programming?

- “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.”

How does GP works?

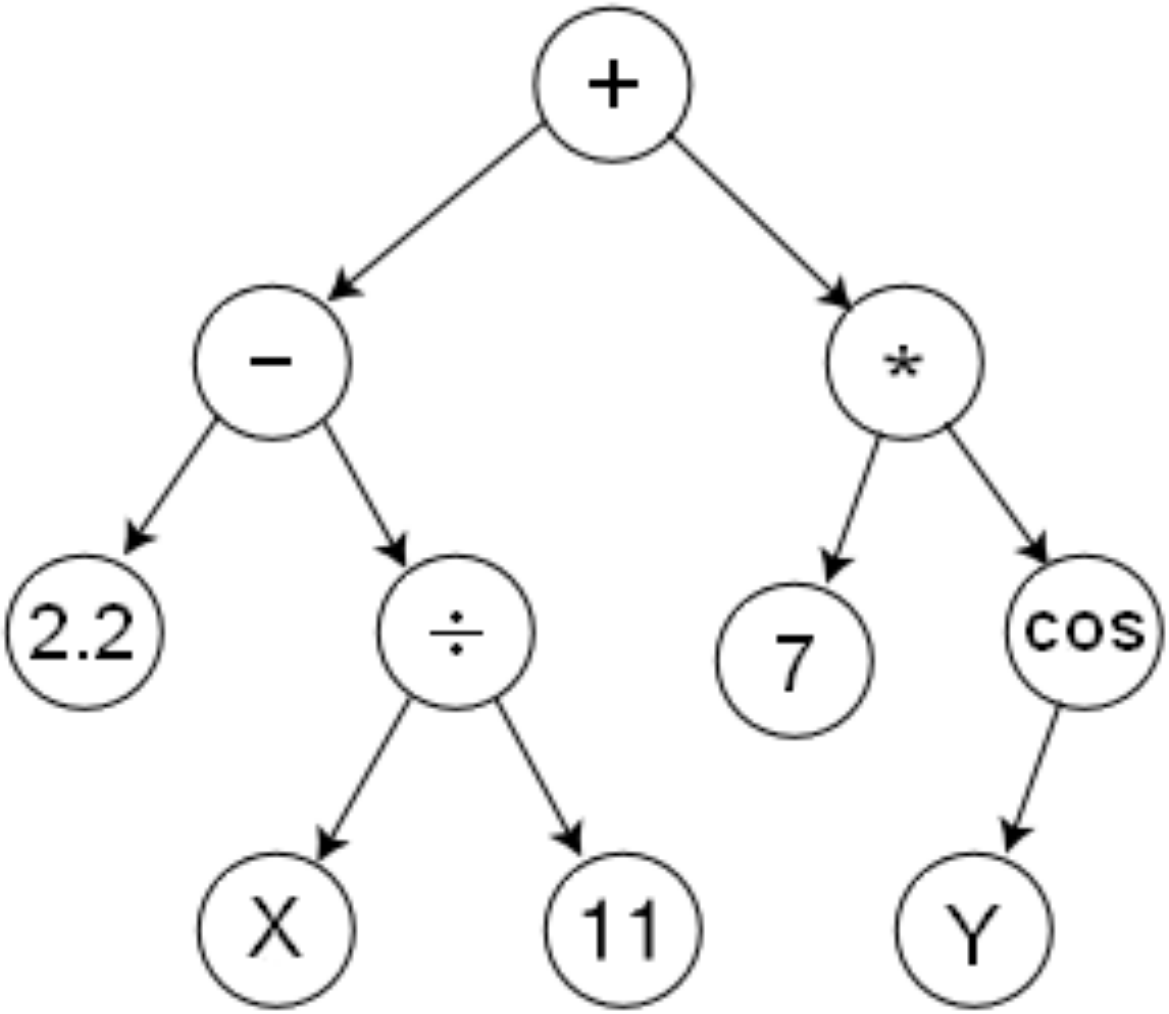
- GP determines how well a program is by running it and then compare it to an ideal (usually the original program). The comparison between the program that was run and the ideal program is expressed in a numerical value called “fitness”. According to the fitness value, the program that does well is chosen 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

GP operators:

- Crossover : works by changing two, or more, programs by combining them together making a new program, just like the biological making of a child.
- Mutation: 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.

Example 1:

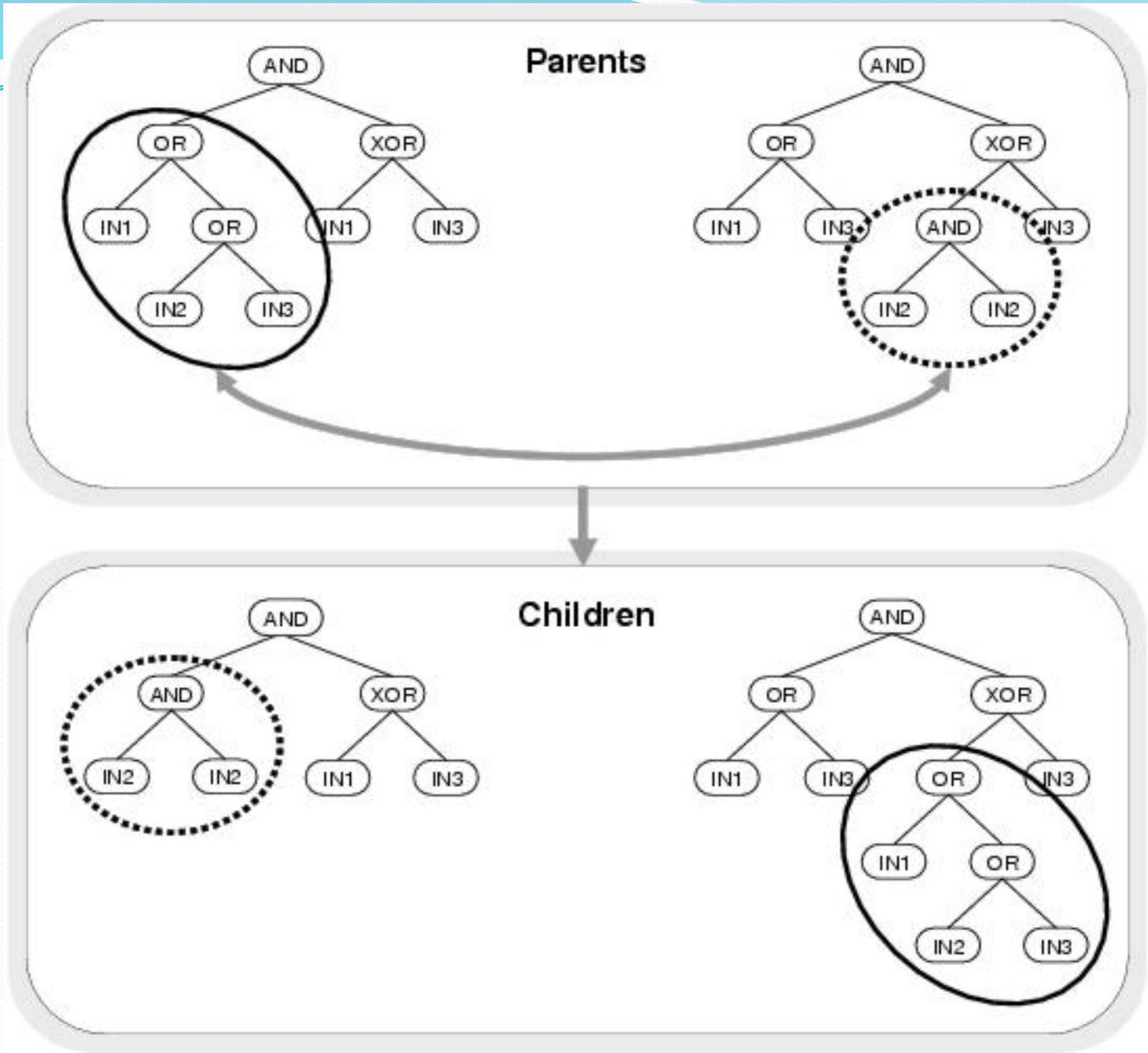
- GP program (2.2 – X / 11 + 7 * COS(Y)) represented in a syntax tree



$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$

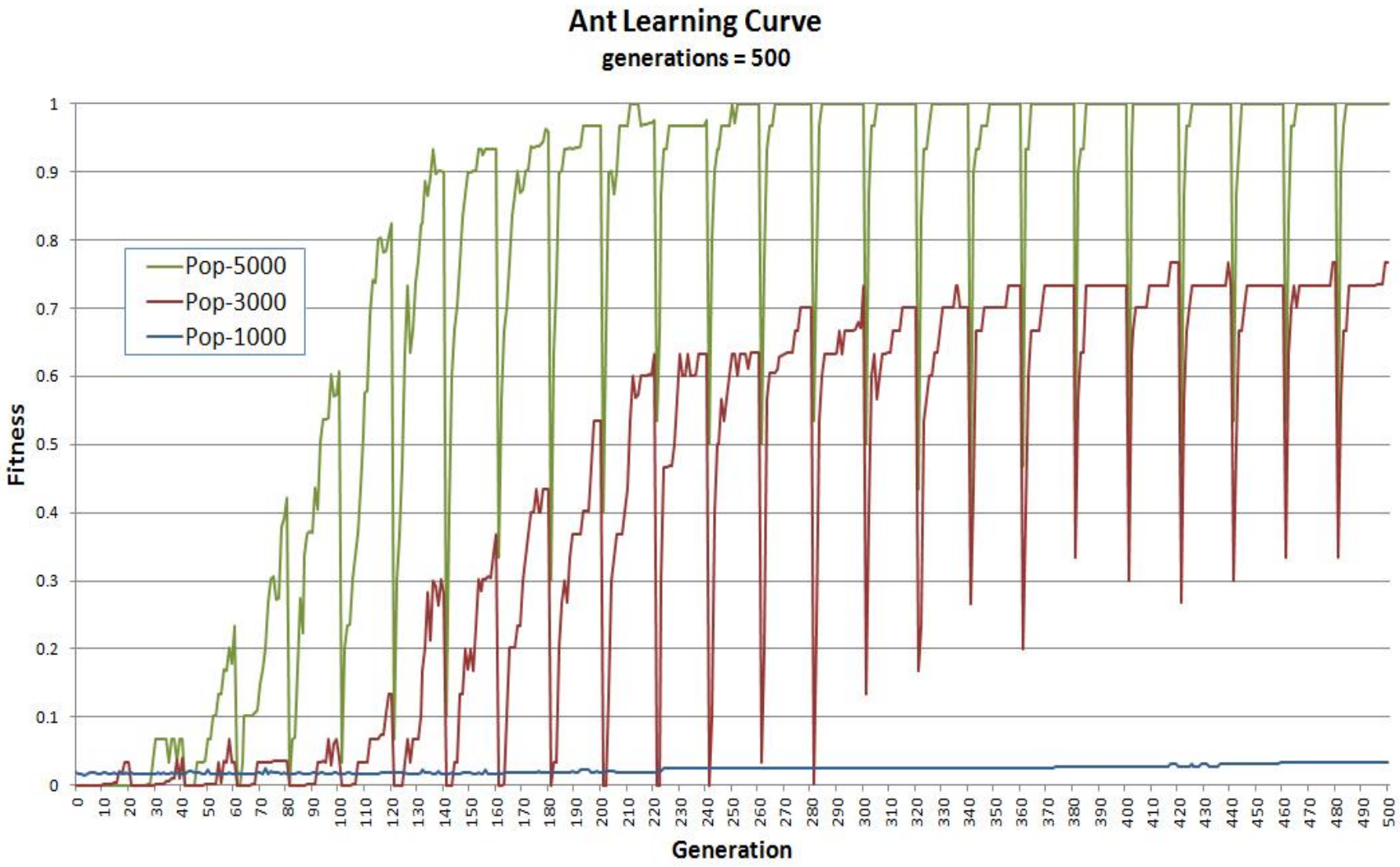
Example 2:

- Sub-trees are selected randomly from two existing parse trees and are then swapped to produce child parse trees:


Parents
AND
OR
XOR
IN1
OR
IN1
IN3
IN2
IN3
AND
OR
XOR
IN1
IN3
AND
IN3
IN2
IN2
Children
AND
AND
XOR
IN2
IN2
IN1
IN3
AND
OR
XOR
IN1
IN3
OR
IN3
IN1
OR
IN2
IN3

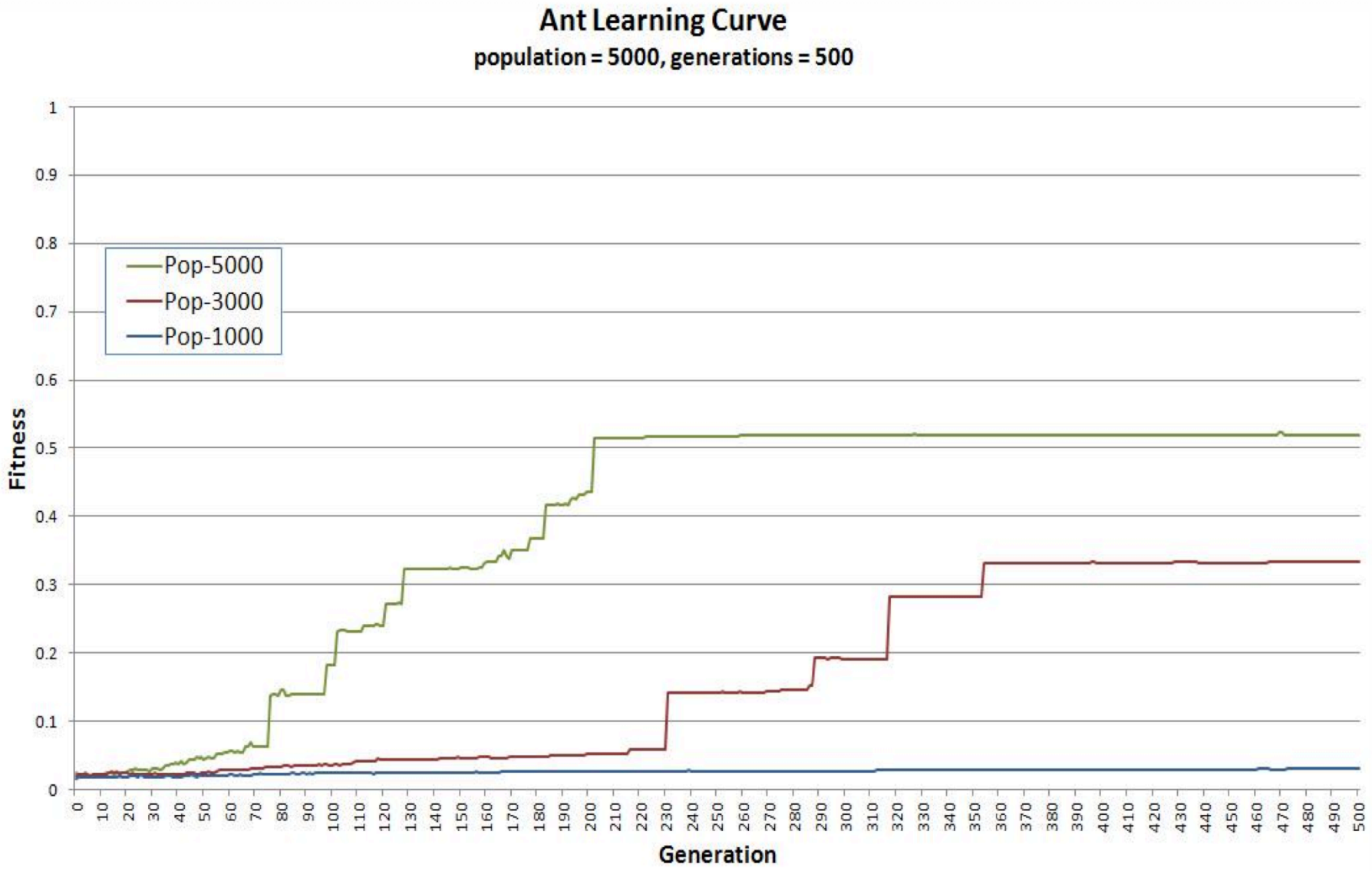
Artificial Ant Experiment:

- A grid with a trial of food pellets.
- Name off trail is “Santa Fe”
- The fitness is measured by the number of food pellets run over by the ant.
- Using 3 population sizes (1000, 3000, 5000)
- Run each population only once at the beginning.
- The graph is:


Ant Learning Curve
generations = 500
Pop-5000
Pop-3000
Pop-1000
Fitness
Generation
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500

Artificial Ant Experiment:

- The experiment run 10 times, each population.
- Graphs:


Ant Learning Curve
population = 5000, generations = 500
Pop-5000
Pop-3000
Pop-1000
Fitness
Generation
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500

Results:

- When the experiment is run once:
- 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
- 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
- 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
- When the experiment is run ten times:

Results:

- 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.
- 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.
- 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.
- 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.
- Large populations run a lot slower in GP.

References:

- 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.
- *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 .