Genetic Programming – Santa Fe Trail Problem

Sanjeev Shrestha

April 06, 2014

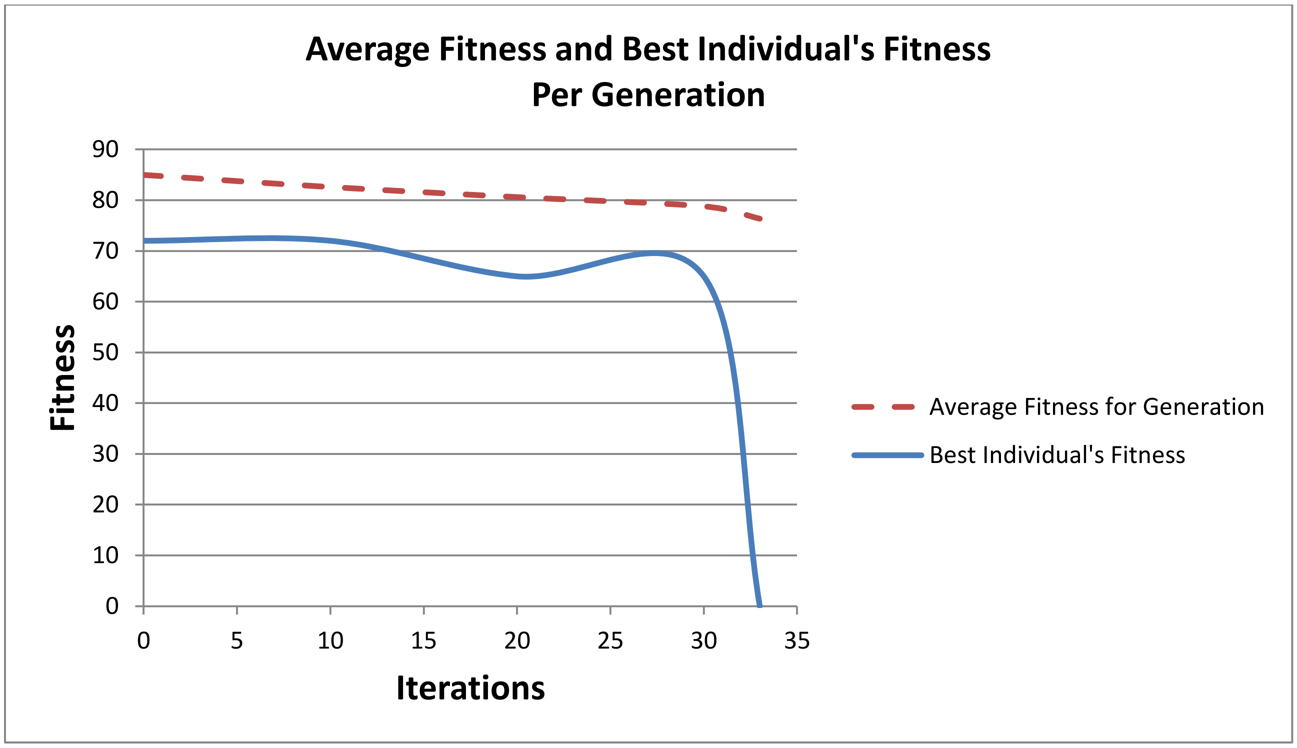
Abstract

Genetic Programming deals with the creation of programs and formulas that serve as a solution to best satisfy the given dataset or the problem. An individual in a genetic program is represented by an expression tree of operators (function set) and operands (terminal set). One of the most interesting problems that is often used as a benchmark in genetic programming is the Santa Fe Trail problem. The classic Santa Fe Trail consists of a 32 X 32 grid board in which the 89 food pellets are lined up randomly making a 144 length trail. The trail consists of spaces and turns in between the food pellets. The problem is defined as an artificial ant must traverse this board and eat all the food in the board. The artificial ant is only given 600 time units up to which the ant must traverse and collect all the food pellets. The project requires us to evolve a path followed by the ant in such a way that it is able to collect all the food, make necessary turns, stay within the bounds of the board and is within the 600 time units. The individuals are subjected to the evolutionary process using genetic operators such as selection, mutation and crossover. This report provides a summary of how the experiment was conducted and verified as to evolve the best path followed by the ant. The artificial ant walked through the expression tree according to the different terminal nodes it encountered. The terminal nodes were responsible for movement of the ant and its direction (both magnitude and direction). Selection performed on the individuals was the tournament selection operation; in which the best individual was selected from the pool of random individuals having tournament size 6. For removing the population, inverse tournament selection was used where the worst individual would be selected for removal from the population. The crossover used was sub tree crossover where random nodes were selected between two trees and crossed over. The mutation done was Node mutation with 30% probability of each node to be mutated. Node mutation was done with nodes having same arity. PROG3 in the function set was thus not subjected to any node mutation as it was the only one with arity 3. The 90/10 rule was employed for selecting a random node, in which, there was a 90% chance to select Non-terminal nodes and 10% chance for terminal nodes. Fitness of an individual was calculated as number of food pellets (89) minus the number of food eaten. Parsimony pressure was also introduced so as to check the size of the trees and limit code growth. Fitness penalty was given on the basis of a distribution of different sizes of the individual. The trend in average fitness and best fitness of an individual in the case of Santa Fe Trail suggested that the convergence took a longer time for average value. The report entails the actual path and the path followed by the evolved ant. The results are good and have solved the Santa Fe Trail under the 600 time unit constraint. Some Java 2D graphics API was used to make the result visualization more interesting and friendly.

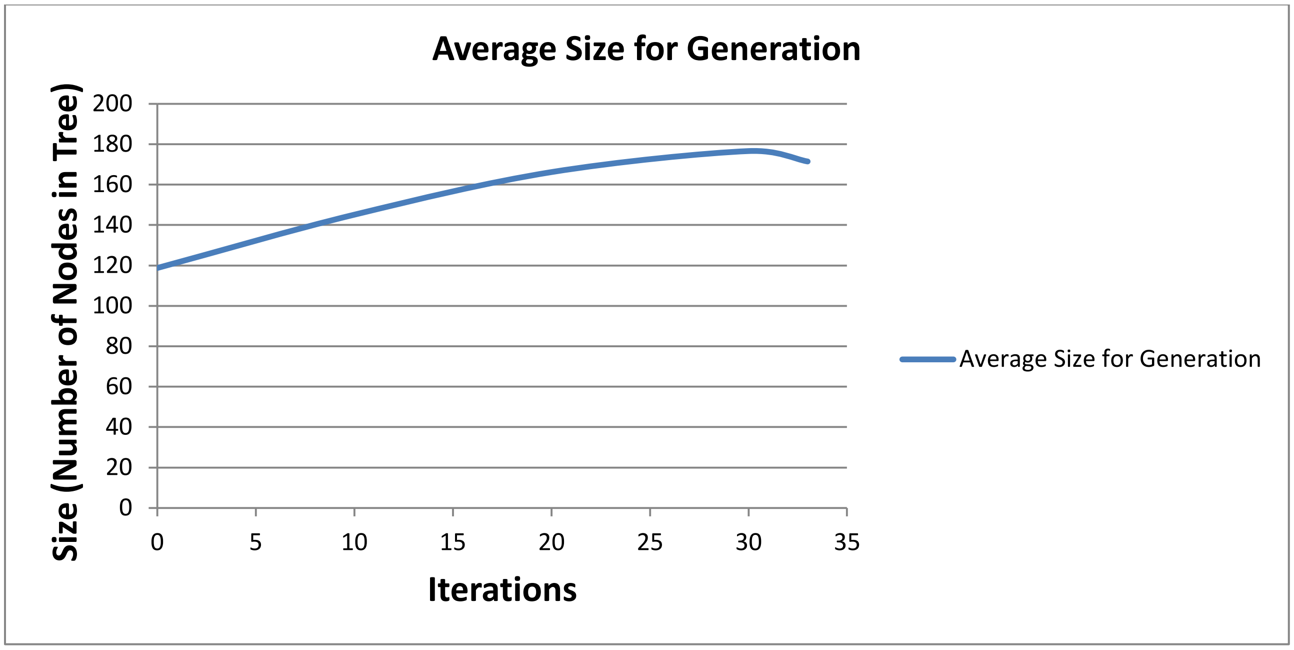
Project Details:-

Algorithm	Steady State Model
Population size	100
Tree Generation Method	Full Method – generates full trees up to given tree depth
Tree Depth	7 – initial tree height
Selection method	Tournament selection (having tournament size 6)
Board Size	An integer array of 32 X 32 filled with NOFOOD = 0 and FOOD = 1
Elitism (if used)	-
Crossover method	Sub tree Crossover
Crossover rate	6% of the population i.e., 6 out of 100 Individuals
Random Node Selection	90/10 rule used (90% nodes selected are Non-terminals and 10% are terminals)
Mutation method	Node Mutation
Mutation rate	30% per node
Operator/non-terminal set	{ PROG3, PROG2, IFFOODAHEAD, IFNOFOODAHEAD }
Terminal set	{ LEFT, RIGHT, FORWARD }
Ant Orientation	Initially EAST, can move to WEST, NORTH or SOUTH depending on command LEFT or RIGHT
Max Arity	3 (as function set has at most 3 branches)
Number of Time Units	600 command evaluations for each ant.
Fitness function	• Fitness = NUMOFFOODPELLETS(i.e. 89) – numFoodEaten • Penalty for Big Trees if (Treesize >= 250) fitness += (int) ((Math.round(size * getFactor(250, size)))); Penalty distribution for getFactor() = { 0.05, 0.07, 0.10, 0.30, 0.50, 0,70, 0.80 } Size range evenly distributed with 10 as step size. Ex: 250 <= size < 260 • Penalty for moving out of bounds Fitness += 8 (constant factor)
Size control (if any)	Parsimony pressure applied for individuals with big size as explained above in fitness function.
Stop Condition	If solution has been found (fitness = 0) or Up to 200 MAX_ITERATIONS

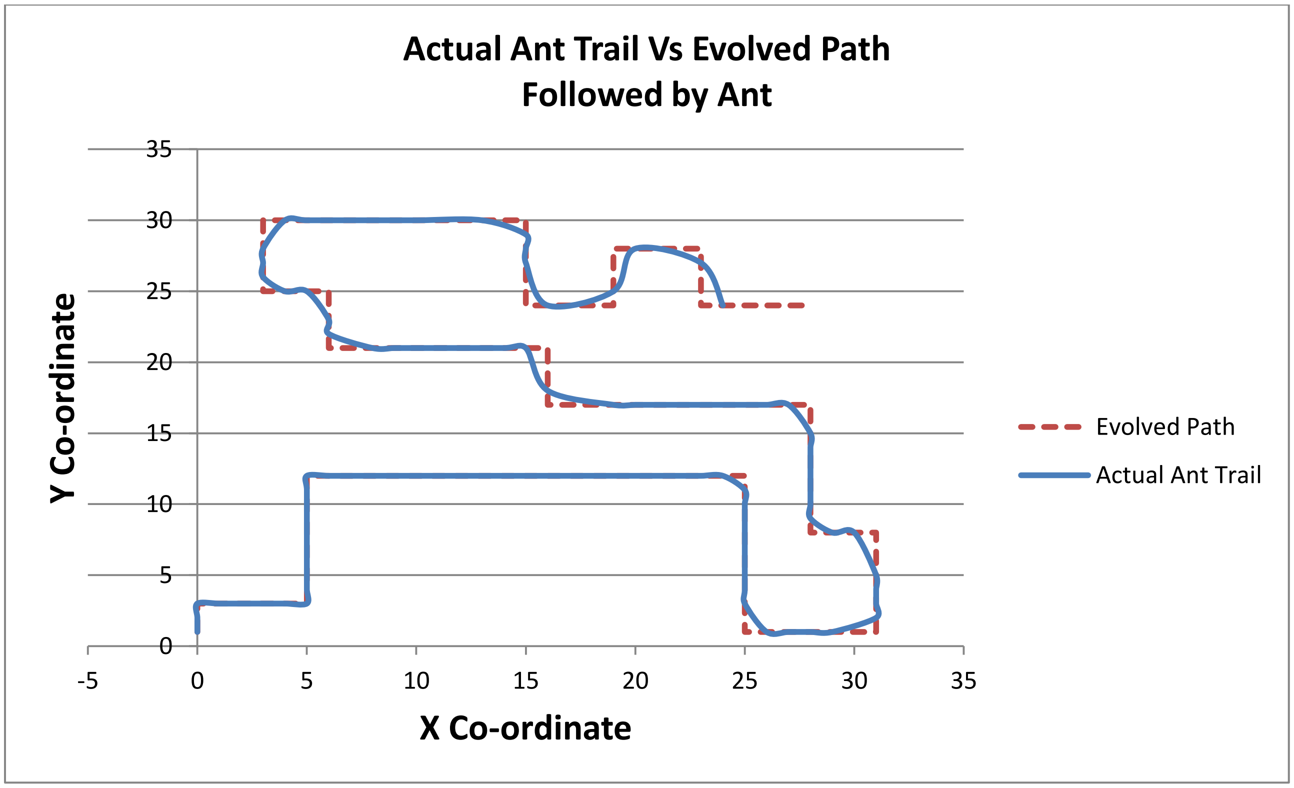
Graph showing average fitness and best fitness evolving over time



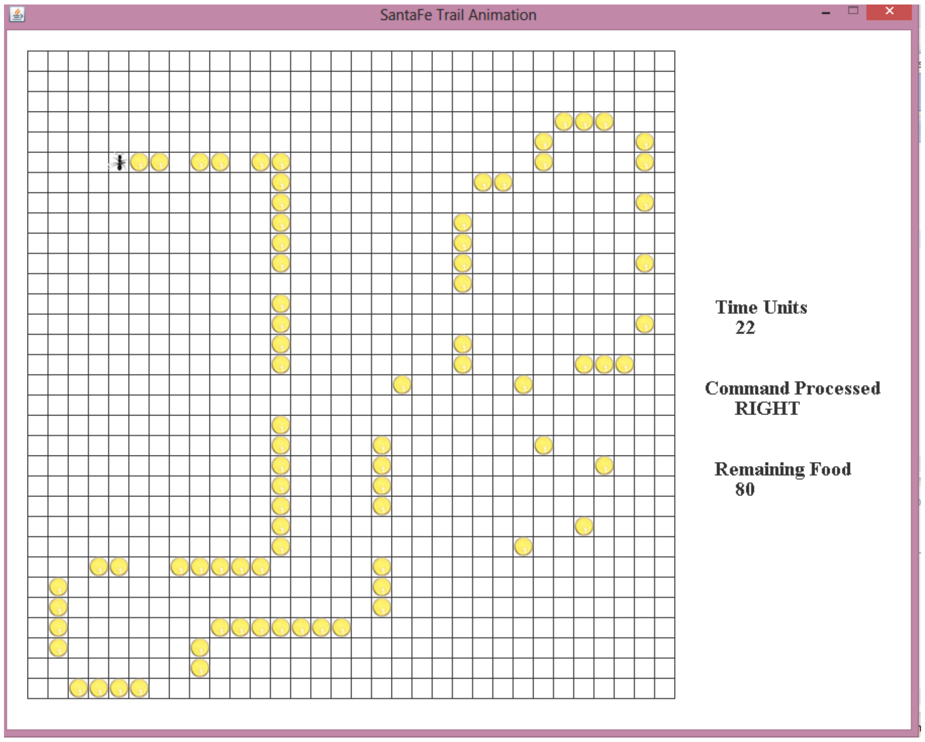
Graph showing code growth in each successive generation



Graph showing Actual Ant Trail and Evolved Path followed by Best Ant



GUI test run for Best Ant



Time Units
22

Command Processed
RIGHT

Remaining Food
80

Discussion

In conclusion, the Santa Fe Trail problem was solved with the help of Genetic programming approach. The approach was slower than a Genetic Algorithm. However, I felt it was faster than the previous symbolic regression problem. Memory and performance issues were faced in this part of the project as well. The iterations became slower due to code growth. Without the help of parsimony pressure the average size of individuals in a particular generation was nearly 400. After using parsimony pressure, the average size of the tree ranged in between 180 to 250. As suggested the trees grew nearer to the point at which parsimony pressure check was applied (i.e. anything above 250 gets a fitness penalty). I also added a penalty of constant value 8 if the ant went out of bounds of the grid. The converging process was slow however; it converged to a good value slowly. The best individual remained constant in a number of iterations. This must be the result of destructive crossovers and mutation. The random number generator was switched from native one to the "MersenneTwister" implementation distributed by Dr. Sean Luke from website http://cs.gmu.edu/~sean/research/mersenne/MersenneTwister.java . After watching a Santa Fe Trail video on YouTube I got the motivation to work on my own GUI implementation. The test run is done on the basis of test results generated after running the GP. Finally, I conclude that the evolved ant eventually collected all the food pellets laid across the trail and the Santa Fe Trail problem was solved using tree based genetic programming approach.