Genetic Programming II: Artificial Intelligence

Handout 11-21-11
Genetic Programming for Artificial Intelligence

• Genetic programming can be used for much more diverse and complicated algorithms than polynomials or the functions arising in symbolic regression.

• To illustrate this, consider the “Artificial Ant” problem.

• This is a good example of artificial intelligence.
1. The ant wants to move along the "Sante Fe" trail (See figure 3.6) and pick up all the available food within the maximum allotted time. The black squares have food and the other squares have no food.
Fig. 3.6 Santa Fe Trail (from Koza book),

Figure 3.6 The Santa Fe trail for the artificial ant problem.
Ant Capabilities

2. The ant can do a limited number of things.

a. It can turn right or left (90 degrees) and look straight ahead to see if there is food on the next square (which counts 1 time period each). 
*Terminal for this is \{right\} or \{left\} if there are turn in place only*

b. It can move straight ahead (which counts as 1 time period). If it moves into a square with food, it eats the food.
*Terminal for this is \{move\}*
• 3. The goal is to develop an algorithm using the actions 2a and b so that the ant accomplishes the goal stated in 1. above.

• **Terminal Set** - is the set of actions: right, left, or move, where “move” means move straight ahead and right or left refer to 90 degree turns.
Functions

- The function PROGNm arg1, arg2, ..., argm first implements arg1, then implements arg2, and so forth until it lastly implements argm. The argi's (i=1,...,m) can be the terminals, right, left or move or other functions.

- The function If food ahead arg1 arg2 we define as “do arg1 if the ant senses that there is food straight ahead and do arg2 if there is no food straight ahead”.

- We let the function set for this problem be

\{If-food-ahead, PROGN2, PROGN3\}

\[
\text{if -food-ahead arg1 arg2 } \Rightarrow \text{ if (food ahead) then arg1 else arg2}
\]

\[
\text{PROGN m arg1, arg2, ..., argm } \Rightarrow \text{ arg1 ; arg2 ; ... ; arg m ;}
\]
• Some examples:

• **PROGN2 (RIGHT) (LEFT)** does what? Will it be successful at getting reaching the goal?
  
  does right; left, only turns in place so nothing happens

• What about **IF-FOOD-AHEAD (RIGHT) (MOVE)**? How much food will it get? Why?
  
  turn away from visible food (to the right) and move. Might get food if there was turn to the right

• How much food would **IF-FOOD-AHEAD (MOVE) (MOVE)** obtain?
  
  =) unknown, depends on if then’s food ahead, and how “deep” the food line runs
FITNESS

• We need a fitness measure. We will use the following

• **Standardized fitness:** We will use the amount of food NOT eaten in the allotted amount of time (400). Hence if all the food is eaten, the fitness is 0, which is the optimal value. ⇒ minimization problem

• Each terminal (right, left, move) counts as one time step. We limit the time steps to 400.

• To evaluate the fitness for a given member S of the population of programs, you do a simulation of the ant moving through the grid using the program S to control its actions.

• In the figures, the X’s show the path of the artificial ant.
# Koza Summary Table

Table 7.3 Tableau for the artificial ant problem for the Santa Fe trail.

<table>
<thead>
<tr>
<th>Objective:</th>
<th>Find a computer program to control an artificial ant so that it can find all 89 pieces of food located on the Santa Fe trail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set:</td>
<td>(LEFT), (RIGHT), (MOVE).</td>
</tr>
<tr>
<td>Function set:</td>
<td>IF-FOOD-AHEAD, PROGN2, PROGN3.</td>
</tr>
<tr>
<td>Fitness cases:</td>
<td>One fitness case.</td>
</tr>
<tr>
<td>Raw fitness:</td>
<td>Number of pieces of food picked up before the ant times out with 400 operations.</td>
</tr>
<tr>
<td>Standardized fitness:</td>
<td>Total number of pieces of food (i.e., 89) minus raw fitness.</td>
</tr>
<tr>
<td>Hits:</td>
<td>Same as raw fitness for this problem.</td>
</tr>
<tr>
<td>Wrapper:</td>
<td>None.</td>
</tr>
<tr>
<td>Success predicate: An S-expression scores 89 hits.</td>
<td></td>
</tr>
</tbody>
</table>
Figure 7.12 is one of the solutions obtained, which Koza calls the “quilter”. Its program is

- \text{PROGN3} \ (\text{RIGHT})
- \text{(PROGN3} \ (\text{MOVE}) \ (\text{MOVE}) \ (\text{MOVE})\text{)}
- \text{(PROGN2} \ (\text{LEFT}) \ (\text{MOVE})\text{)}

You should understand this and be able to indicate how the ant will move given an expression like this.

\(\Rightarrow \text{ right, move } x3, \text{ left, move}\)

Note that for all “solutions”, once the ant moves you start again at the beginning of the program.

at the end of the program execute, return to program start
Path of the Quilter

Figure 7.12 Path of the quilter from generation 0 of the artificial ant problem.
Figure 7.13  Path of the looper from generation 0 of the artificial ant problem.
The “looper” (Figure 7.13 finds the first 11 pieces of food and then goes into an infinite loop when it encounters a NEW thing—a gap in the food trail.
Path of Avoider

Figure 7.14 Path of the avoider from generation 0 of the artificial ant problem.
Figure 7.14 shows the “avoider”

( IF-FOOD-AHEAD (RIGHT)
  (IF FOOD-AHEAD (RIGHT)
    (PROGN2 (MOVE) (LEFT))))
Best of Run Individual for Artificial Ant

Note: This is a parse tree, not an execution flow diagram.

Figure 7.16 Best-of-run individual for the artificial ant problem.
Best Solution in LISP

(IF-FOOD-AHEAD (MOVE)
  (PROGN3 (LEFT)
    (PROGN2 (IF-FOOD-AHEAD (MOVE)
               (RIGHT))
      (PROGN2 (RIGHT)
        (PROGN2 (LEFT)
          (RIGHT))))
    (PROGN2 (IF-FOOD-AHEAD (MOVE)
               (LEFT))
      (MOVE))))).
Figure 7.15

- Figure 7.15 shows the fitness curve and you can see that the average fitness drops steadily (a good thing since we are trying to minimize fitness)
- and the best solution drops slowly until the best solution is found at the end.
- (Recall this is from one run of the GP so the results could be different for other runs.)

```
GA
1. random population
2. fitness eval
3. pick parents
4. generate offspring, goto 2
```
Figure 7.15 Fitness curves for the artificial ant problem.
Best of Run Individual for Artificial Ant

Again, this is a parse tree and not an execution flow.

Figure 7.16 Best-of-run individual for the artificial ant problem.
Here is an interpretation of what this S expression in Figure 7.16 will do:

IF-FOOD-AHEAD senses whether there is any food in the square that the ant is currently facing. If food is present, the left branch of the IF-FOOD-AHEAD test is executed and the ant MOVEs forward. When the ant moves onto a place on the grid with food, the food is eaten and the ant receives credit for the food.

If the IF-FOOD-AHEAD test at the beginning of the S-expression senses no food, the ant enters the three step PROGN3 sequence immediately below the IF-FOOD-AHEAD test.
The ant first turns LEFT. Then a two step PROGN2 sequence begins with the test IF-FOOD-AHEAD.

If food is present, the ant MOVEs forward.

If no food is present, the ant turns RIGHT.

Then the ant turns RIGHT again.

Then the ant (pointlessly) turns LEFT and RIGHT in another two-step PROGN2 sequence.

The net effect is that the ant is now facing right relative to its original facing direction (i.e., its direction at the beginning of the execution of this S-expression).
• The ant next executes the final two-step PROGN2 subtree at the far right of the figure.
• If the ant now senses food via the IF-FOOD-AHEAD test, it MOVES forward.
• Otherwise it turns LEFT. The ant has now returned to its original facing.
• The ant now executes an unconditional MOVE, thereby advancing forward in its original facing direction if it has not found any food to the immediate right or left. (Why does it do this?)
How the Program Functions

• Note that because the state of the ant (i.e. the location of food relative to the location and direction of the ant) changes over time as the ant moves and eats food, different parts of the S-expression are often executed on each evaluation.

• The repeated application of the Algorithm in Figure 7.16 allows the ant to negotiate all the gaps and irregularities of the trail and to eat all the food in the allotted time.
• This is an example of MACHINE LEARNING—the computer generates an option and then evaluates its performance (i.e. computes fitness) and on the basis of this, it decides how to possibly change the S-expression.

• What would happen to the best solution if you shortened the allotted time?
The Program Responds to Situations that Occur along the path rather than to specific paths

- Recall that the ant will always look in the direction it is facing and will detect if there is food in the space ahead that it is facing.
- What are the different types of situations the ant encounters?
  - Without moving right or left, ant sees food ahead. (occurs in first step) What does it do?
  - What other situations does it encounter? (See Board)
Board: Other issues for the Ant

- In order that it occurs
  1. Right corner with no food gap
  2. Left corner with no food gap
  3. One Food gap on straight line
  4. Two adjacent food gap on straight line
  5. One Food gap on the corner
  6. 2 food gap starting before the left corner
  7. 3 food gap starting before right corner
  8. 3 food gap starting before left corner
Applying Program Solution in Fig. 7.16 to a new trail

• What if you developed a new trail (different location of corners, different numbers of boxes in a straight line between corners), but it only contained the specific problem characteristics of the Sante Fe problem (e.g. single, double and triple gaps in food along a straight line or before turning a corner).

• How well would your algorithm in Fig 7.16 work on this new trail? Explain.

• (Would it work at all or work perfectly or would it be sub optimal?)
Power of GP for Artificial Intelligence

• So these results are remarkable.
• We have an program that will work will on all situations that have the same characteristics.
• In other words our program can have the autonomous vehicle (an “ant” in this case) ADAPTS to the different situations (multiple gaps, corners, etc.) whenever it encounters them.
• The GP optimization was not for a specific trail but rather for the set of situations the vehicle would encounter.
• What would happen if you expected the ant to be able to perform well on a whole series of trails, some of which have different characteristics (i.e. three empty squares at a corner as in Figure 7.17)?

• How would you compute this?

• Would the solution change?

• Is the optimal S-expression likely to become longer or shorter?
Applying the Program to a trail different from the Sante Fe Trail

So we can ask, how well could GP work on a variety of applications?

• What if we had another trail. Would the same “program” work?
• What if you only two steps to the first corner. Would your program still be able to work?
• What if the first corner goes left instead of right?
• What if the first food gap is replaced with food?
• What if you had two empty boxes before turning a corner?
• How would you tell if the old program would work on a new trail?
What are the Characteristics of the Problem?

• How would you tell if the old program would work on a new trail?
• You goal is to develop a program that will work on problems with certain characteristics?
• What are the characteristics of the Sante Fe Trail and how might they be different for other problems?
Autonomous Vehicle

• The artificial ant is a simplified version of the problem of controlling an autonomous vehicle.
• Autonomous vehicles (AV) are used for moving without human intervention and need to be able to respond to what they “see” or detect, including obstacles or the presence of something they need to collect.
• There are autonomous vehicles for cars/trucks; boats/submarines; tiny airplanes, Planetary Exploration.
• These AV have many applications including military or hazardous operations where we don’t want to risk human life.
• Examples: NY City Water supply; desert race with AV, tracking oil slicks, Mars Exploration
• Hence this is a very important application. Is GP the only way to do this??
What Does GP have to find for Autonomous Vehicles (AV)?

- We do not want to program an AV just for one specific course (since there are easier ways to do that like go 3 steps forward, turn right, go 5 steps, turn left, etc.—like MapQuest).
- Instead, the goal is to put the AV on a number of different courses and have it succeed based on what it sees or otherwise senses.
- Most AV have cameras and/or other sensing devices.
Are any of these gaps going to cause the Ant to go in the wrong direction with the previous program?

Figure 7.17 The Los Altos Hills trail for the artificial ant.