Genetic Programming

Genetic Programming

- Genetic Programming applies the concept of Genetic Algorithms to Automatic Programming.
- Automatic Programming refers to algorithms that generate a computer program to do a specific task.
- Why would you want to use Automatic Programming?

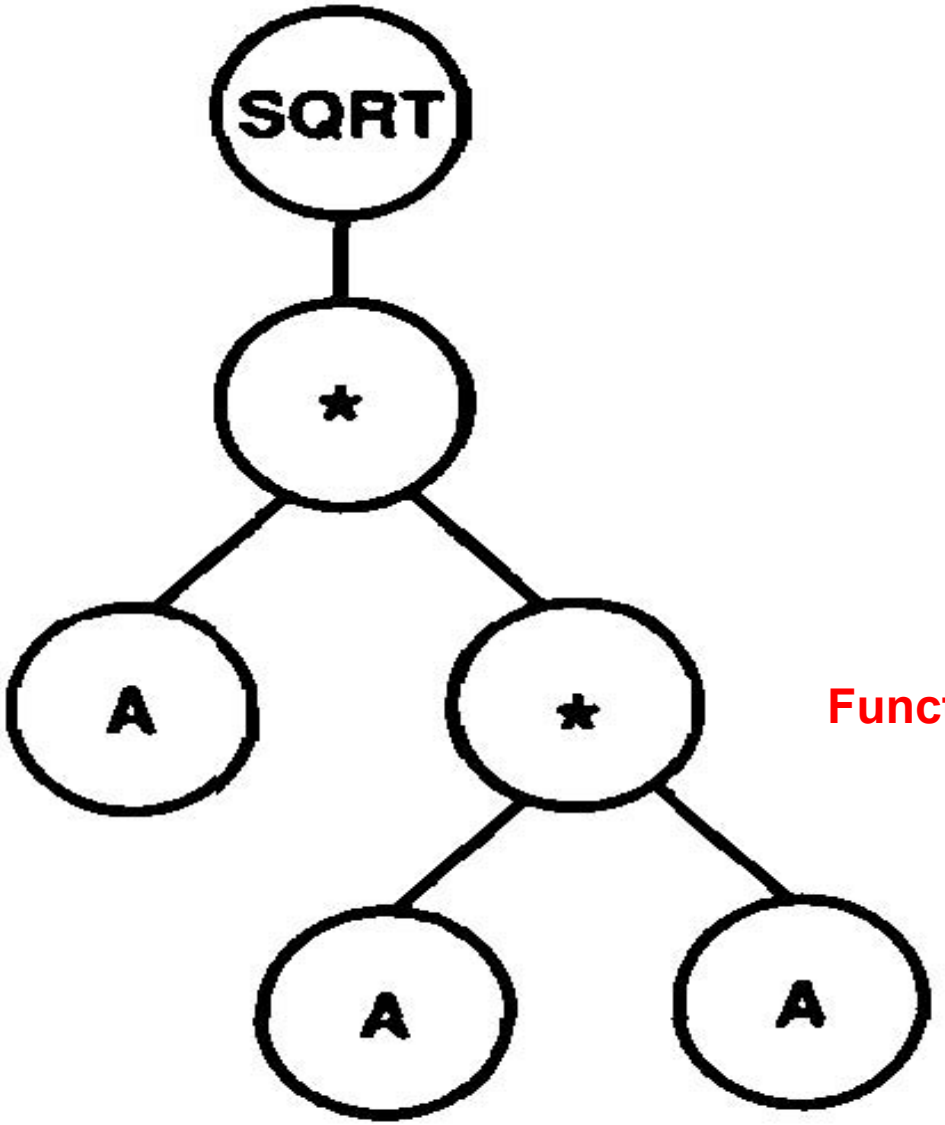
- References to this area include:
- Koza, John (1992) *Genetic Programming* MIT Press
- Mitchell, Melanie (1996) *An Introduction to Genetic Algorithms*, MIT Press (pp 35-53)

First Example: Using Genetic Programming to find Relationship between radius and orbit of planet

- Suppose that you want to find the relationship between the average **distance** of a plant from the sun (A) and the orbital **period** of the planet (P).
- You are given the following data

Planet Name	A	P
Venus	.72	.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.9
Saturn	9.23	29.4
Uranus	19.1	83.5

Genetic Programming uses Parse Trees rather than binary strings for decision variables



Function

Terminal

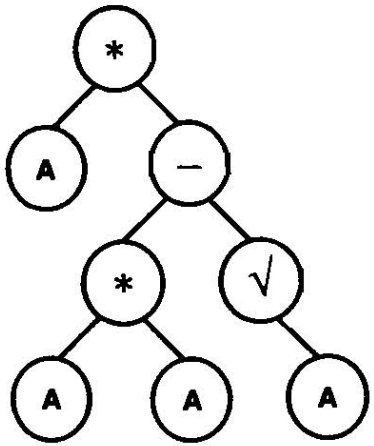
Example of a parse tree for $(A^3)^{1/2}$ = SQRT(*A(*AA))) (*Lisp*)

This is an example of simple program

Contents of Parse Tree

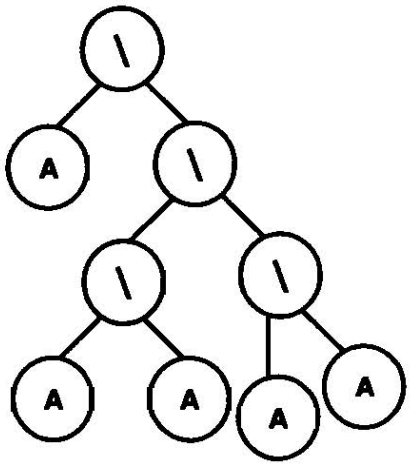
- The tree consists of
- **Functions**- the set of commands available for building solution expressions
- **Terminals**- the set of data (includes decision variables) for the solution expression
- The following shows more examples of parse trees representing a possible solution to the problem.
-

Three possible solutions to the relationship between P and A



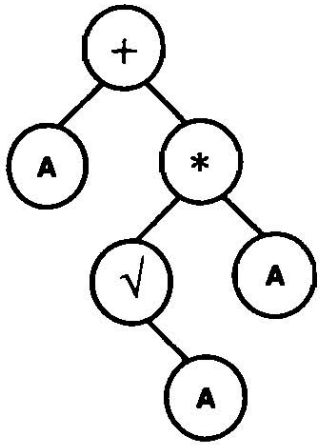
A * [(A * A) – √ A]

fitness → = 1



A \ [(A \ A) \ (A \ A)]

= 3 = fitness



A + (√ A * A)

fitness = 0

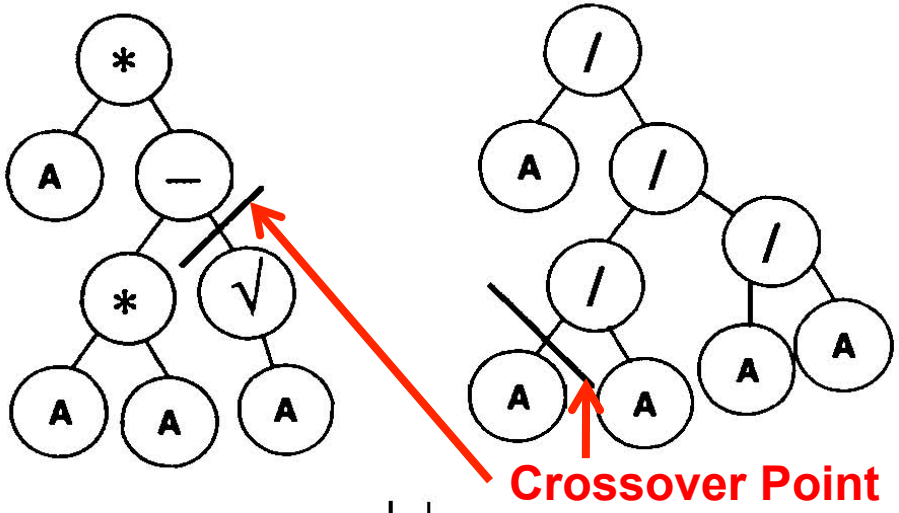
Fitness Cases:

Planet	A	Correct Output (P)
Venus	0.72	0.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.9
Saturn	9.53	29.4
Uranus	19.1	83.5

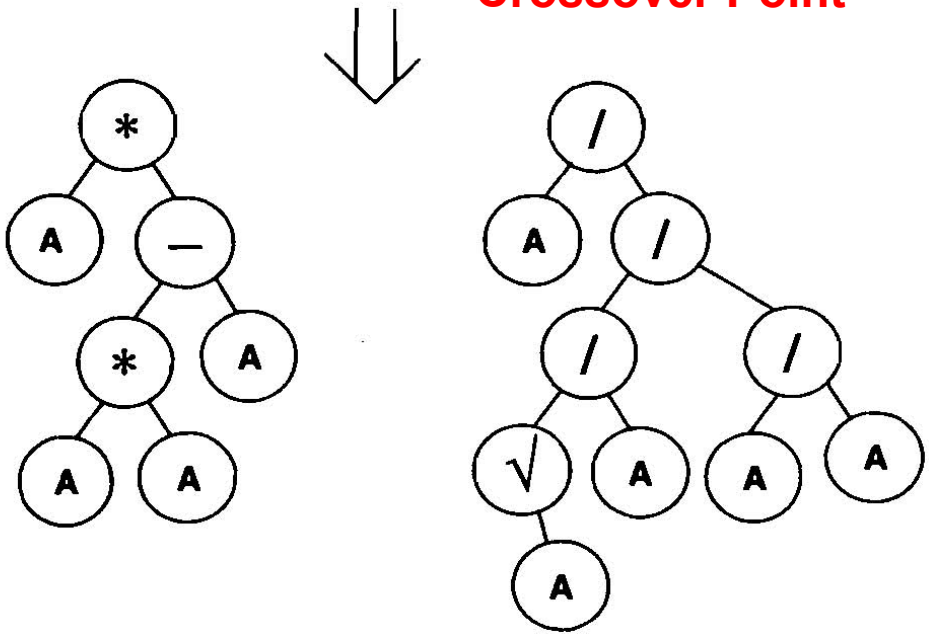
fitness depends on $f_i(A) - P$

Crossover in Genetic Programming

Parents



Offspring



Genetic Programming is …

- Essential aspects of GP
  - Dynamic length structures
  - Evaluate by Execution
  - Syntax Preserving Operators
- Non essentials of GP
  - Parentheses (Lisp Syntax)
  - Lisp Commands
  - Tree Structures

How Different from GA?

end handout 11-16-11

Koza's Algorithm

- Step 1
- **Choose a set of possible functions and terminals for the program.**
- *You don't know ahead of time which functions and terminals will be needed.*
- *User needs to make intelligent choices for best GP performance.*
- *For planetary orbital problem we guessed that the function set is {+, -, *, /, sqrt) and the terminal set is A. (If you add more functions and terminals, the problem takes longer to compute a good answer.)* *handout 11-18-11*

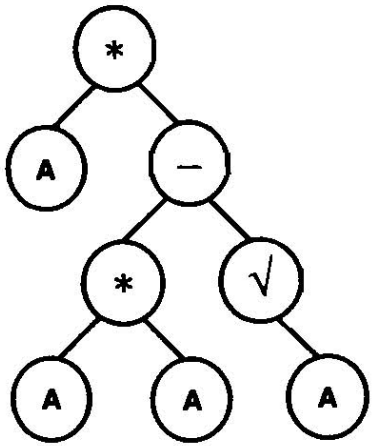
Step 2

- **Generate an initial population of random trees (programs) using the set of possible functions and terminals.**

- *Random trees must be syntactically correct programs—the number of branches extending from each function node must equal the number of arguments taken by that function. (Three such random trees are given in Fig. 2.2.)*

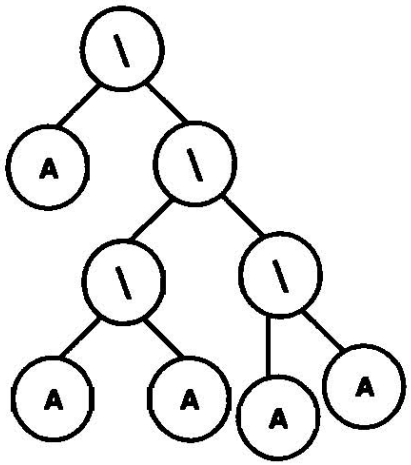
- *Notice randomly generated programs can be of different sizes (i.e. can have different numbers of nodes and levels in the trees.)*

Repeated Slide: Three possible solutions to the relationship between P and A



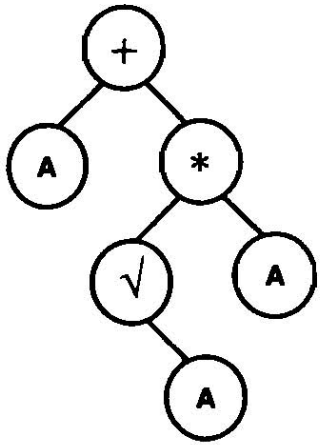
A * [(A * A) – √ A]

f = 1



A \ [(A \ A) \ (A \ A)]

f = 3



A + (√ A * A)

f = 0

Fitness Cases:

Planet	A	Correct Output (P)
Venus	0.72	0.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.9
Saturn	9.53	29.4
Uranus	19.1	83.5

Koza's Algorithm (continued)

- Step 3
- **Calculate the fitness of each program in the population by running it for a set of "fitness cases" (a set of inputs for which the correct output is known).**

- *Fig. 2.2 gives the fitness for the three random programs generated.*

- *The difference in fitness among members of the population are basis for "natural selection"*

BOARD

- Notice– have to run each program in the population.
- If the programs are complex, this could be extremely expensive.
- Most of the examples you will see are rather simple programs, but the equations in the programs are computed by the genetic programming.

Step 4

- **Apply selection, crossover, and (perhaps) mutation to the population of random program to form a new population.**

- *Recommended that 10% of trees in population (chosen probabilistically in proportion to fitness) are copied without modification into the new population (How related to elitism?)*
- *The remaining 90% of the new population is formed by crossovers between the parents selected (in proportion to fitness) from the current population.*
- *Fig. 2.3 shows an example of crossover.*

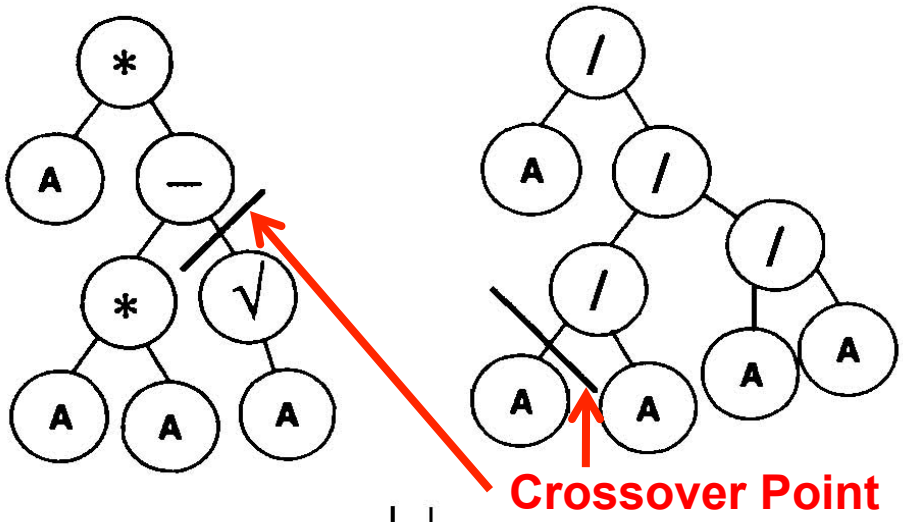
Crossover

- *Crossover allows the size of a program to increase or decrease. (In cross over, all information from the cross over point to the terminal is incorporated into the crossover.)*
- *Mutation can be performed by choosing a random point in a tree and replacing the subtree beneath that point by a randomly generated subtree.*
- *Koza typically does not use a mutation operator.*

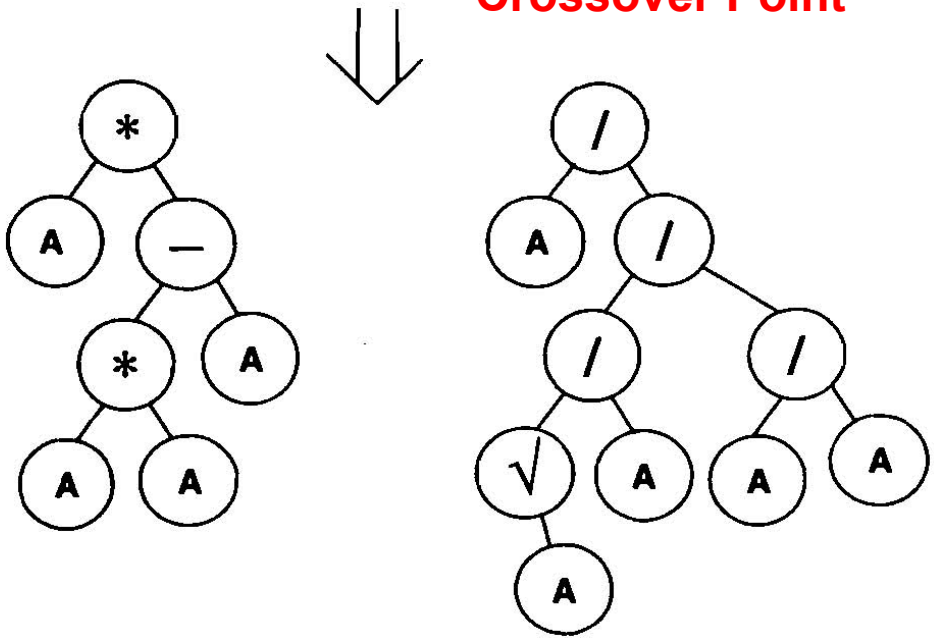
- ***Step 3 and 4 are repeated until a stopping criterion is satisfied.***

Repeated Slide: Crossover in Genetic Programming

Parents



Offspring



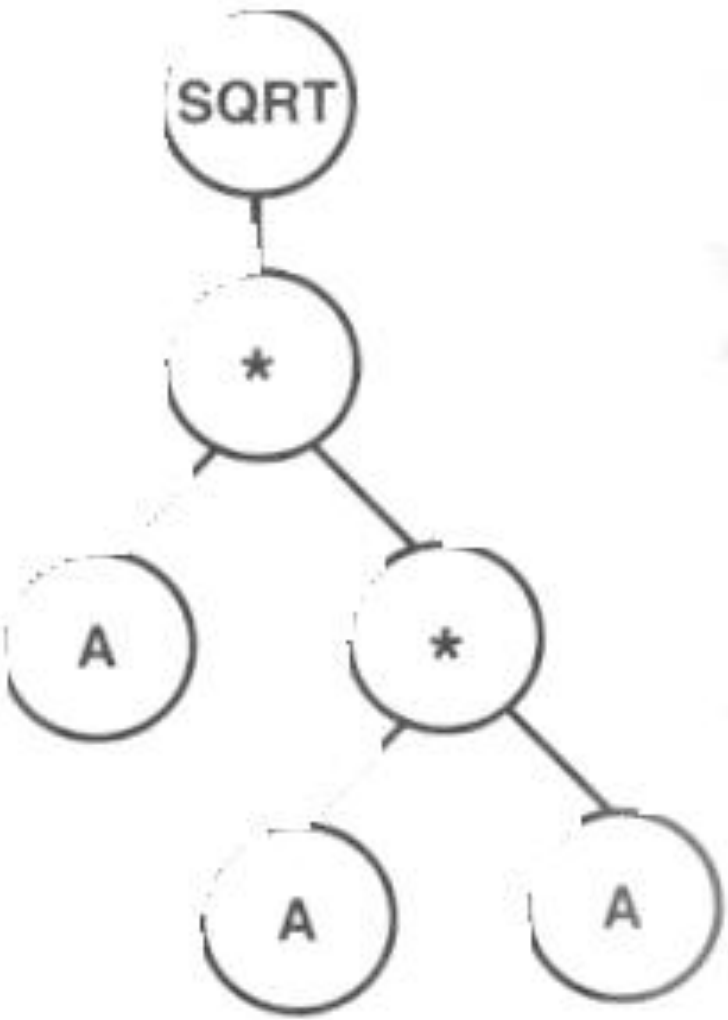


Figure 2.1 Parse tree for the Lisp

- **Best Solution Obtained Earlier for the Relationship between P and A:**

$P = (A^3)^{1/2} =$

SQRT(*A(*AA))) (Lisp)

Was the GP solution of $P = sqrt(A^3)$ Correct?

- Kepler's Third Law says $P^2 = cA^3$ based on physics and mathematics.
- And units can be selected so that $c = 1$ so
- $P = sqrt(A^3)$
- The Genetic Program found this as the best solution after some generations.
- *So does that mean we don't need theoretical thinkers?*

Symbolic Regression

- Genetic Programming can be applied to many types of problems.

- One of these areas is what Koza called "**symbolic regression**" (e.g. the orbital Problem).

- You are familiar with the idea of <u>regression</u> in statistics.
  For example, if you speak of linear regression, you want to compute the coefficients a, b, c, …, z such that
  $Y = a\,x + b\,x^2 + \ldots + z\,x^n$

- For nonlinear regression, the expressions on the right side of the equation can be sums of nonlinear terms like polynomials, <u>but you assume that you know the form of the equation in regression and that you are looking only for the coefficient values.</u> (Some of the coefficients might be zero, thereby dropping some terms.)
-

  So nonlinear regression does not include some types of regression equations that could be included in symbolic regression
- **For example $f(x)= \sin(x)/(\cos(x) + 12x + x^2$**

board

- Write polynomial on board and indicate the coefficients.

- What if the formulae is something like
- P^2=A^3—could you have found that by regular regression—yes but you would need to have know to make the necessary log transformation
- logP= C logA
- And regression would give C=3/2 and logP=(3/2) logA
- However, there could have been many other forms and you would not have known how which transformations to make .
- In many cases there are no possible transformations (e.g. . Y=(cos (x)/(1+x^.5))^4.5)

GP Symbolic Regression Selects the Form as well as coefficients

- In **symbolic regression**, you are trying to find the ***form*** of the equation as well as the coefficients. This is a much harder problem.
- This is <u>data to function regression.</u> It can be done in general for multidimensional input and output.
- Genetic Programming can be used to solve symbolic regression problems

Symbolic Regression Problem

- For this first example we will take a one dimensional example so x and y are scalars.
- Assume we are given 20 pairs of data (x_k, y_k) k=1,…,20.

We want to find a function f such that

$Y = f(x) + error$ with |error| less than .01

First Steps

- In general we do not know the mathematical expression for f.
- However, in this example we are going to use 20 pairs of data (x_k ,y_k) where the value of y_k has been generated from

  $y_k = (x_k)^4 + (x_k)^{3+} (x_k)^2 + (x_k) + 1$
- Hence the best expression for f is this 4th order polynomial (but the GP user does not know this.)
- We will see if the genetic programming approach will discover that this is the best form for f. (The GP does not know before it starts calculating that a 4th order polynomial is the form of f(x) = Y.)

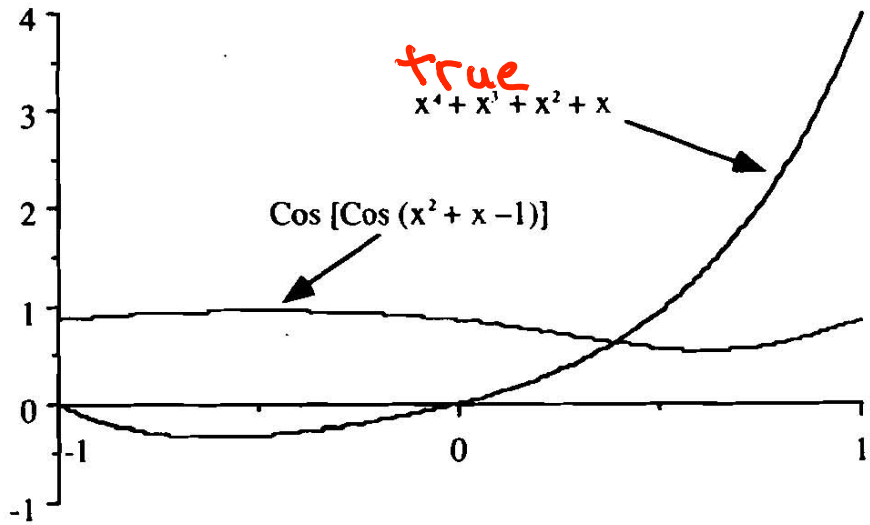
Fitness in Symbolic Regression Example

- Fitness is calculated by comparing the value of the true function y(x) to the population member $s^k(x)$ at twenty points $\{x_j\}$ between 0 and 1 . So fitness is

$$fitness(s^k) = \sum_{j=1}^{20} | y(x_j) - s^k(x_j) |$$

Table 7.4 Tableau for the simple symbolic regression problem.

Objective:	Find a function of one independent variable and one dependent variable, in symbolic form, that fits a given sample of 20 (x_i, y_i) data points, where the target function is the quartic polynomial $x^4 + x^3 + x^2 + x$.
Terminal set:	X (the independent variable).
Function set:	+, −, *, %, SIN, COS, EXP, RLOG.
Fitness cases:	The given sample of 20 data points (x_i, y_i) where the x_i come from the interval $[-1, +1]$.
Raw fitness:	The sum, taken over the 20 fitness cases, of the absolute value of difference between value of the dependent variable produced by the S-expression and the target value y_i of the dependent variable.
Standardized fitness:	Equals raw fitness for this problem.
Hits:	Number of fitness cases for which the value of the dependent variable produced by the S-expression comes within 0.01 of the target value y_i of the dependent variable.
Wrapper:	None.
Parameters:	$M = 500$. $G = 51$. M = population ; G = generation
Success predicate:	An S-expression scores 20 hits. = solution is within .01 of data for all points



Generation 0 two examples of individual "programs"=functions

Figure 7.22 Median individual from generation 0 compared to target quartic curve $x^4 + x^3 + x^2 + x$ for the simple symbolic regression problem.

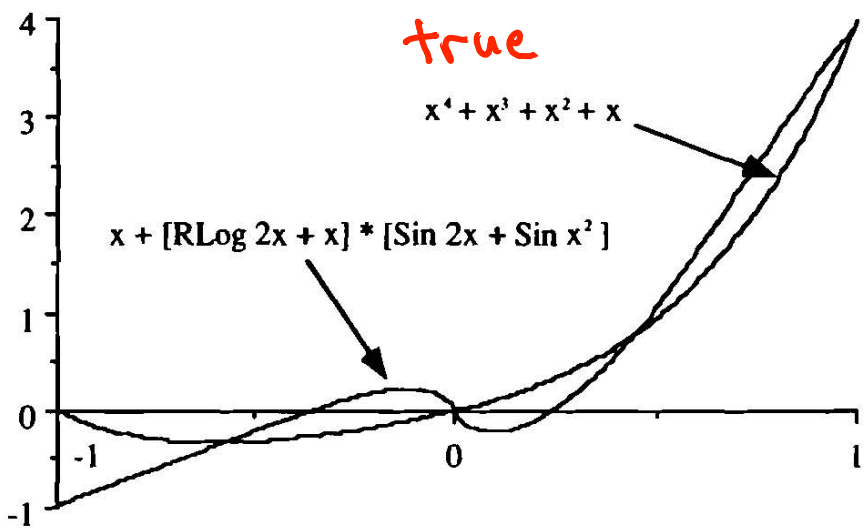


Figure 7.23 Second-best individual from generation 0 compared to target quartic curve $x^4 + x^3 + x^2 + x$ for the simple symbolic regression problem.

Best of Generation from Generation 0

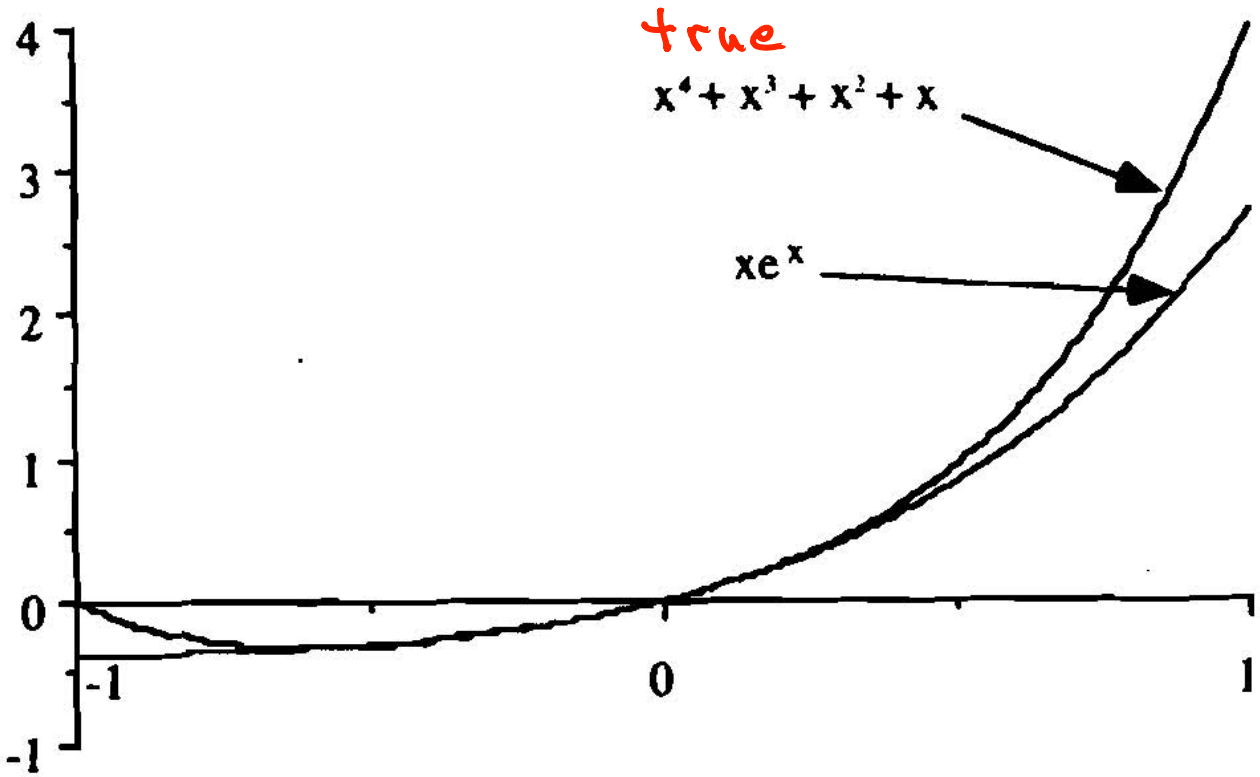


Figure 7.24 Best-of-generation individual from generation 0 compared to target quartic curve $x^4 + x^3 + x^2 + x$ for the simple symbolic regression problem.

Results for Best of generation individual from generation 0

Table 7.5 Simplified presention of the simple symbolic regression problem with only five fitness cases.

1	x_i	-1.0	-0.5	.00	$+.5$	$+1.0$
2	$y = xe^x$	$-.368$	$-.303$	.000	.824	2.718
3	T	0.0	$-.312$	.000	.938	4.0
4	$\lvert T - y \rvert$	.368	.009	.000	.113	1.212

T is true function. Y here is best of generation 0. You compute fitness by adding all the numbers in 4th row (but there would be 20 values not just the 5 values shown here)

Best of Generation 2

- $X^4 + 1.5x^3 + 0.5x^2 + x$

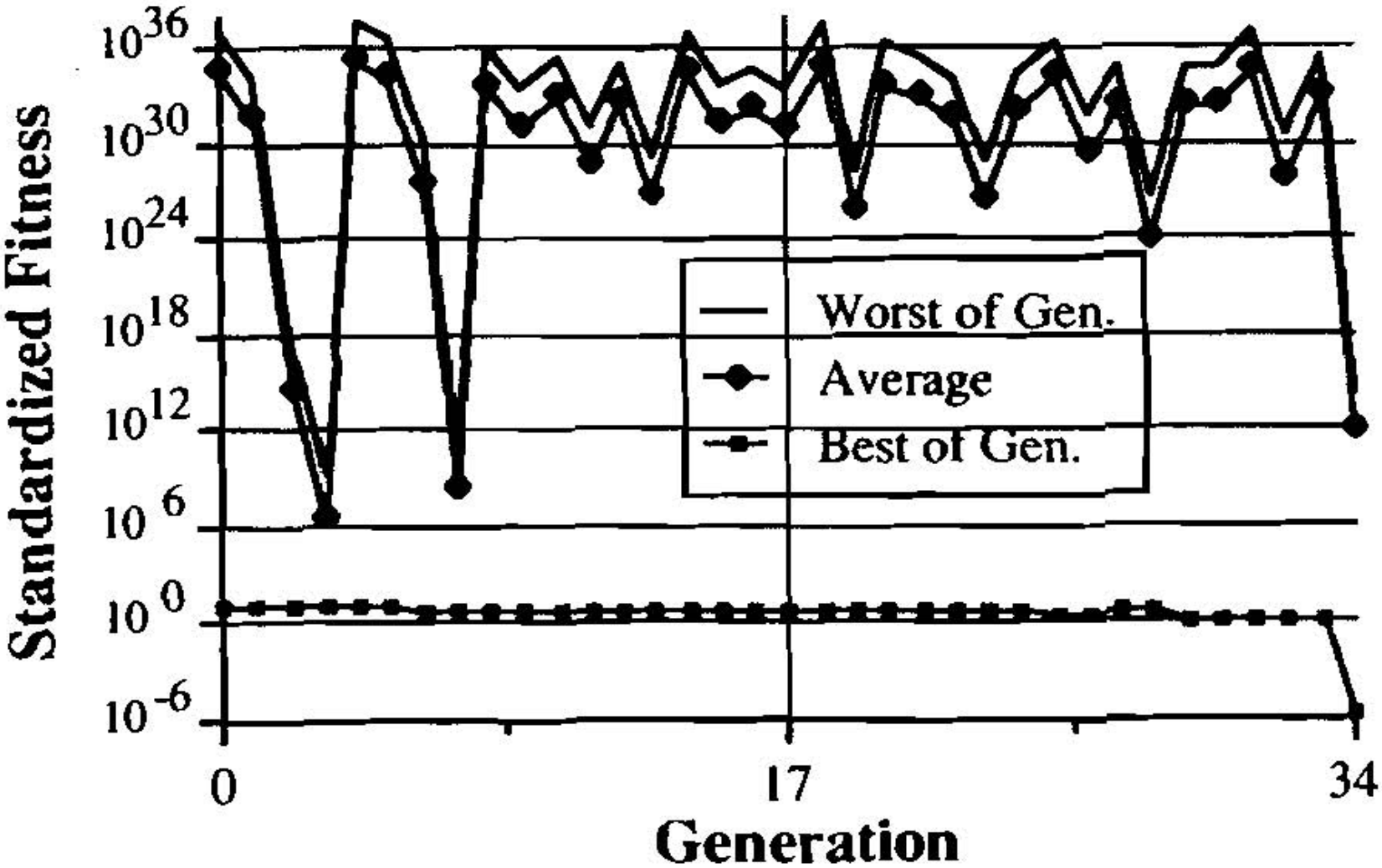
- Raw fitness is 2.57 (improved from 4.47 in generation 1 best individual)
- Notice that we have coefficients. How were they generated given that we only started with coefficients of 1?

BOARD

- How to make coefficients:

- $(X+x)/x=2$
- How fast can you create the right coefficient (easier if number is 2 than if the number is 2.1987.

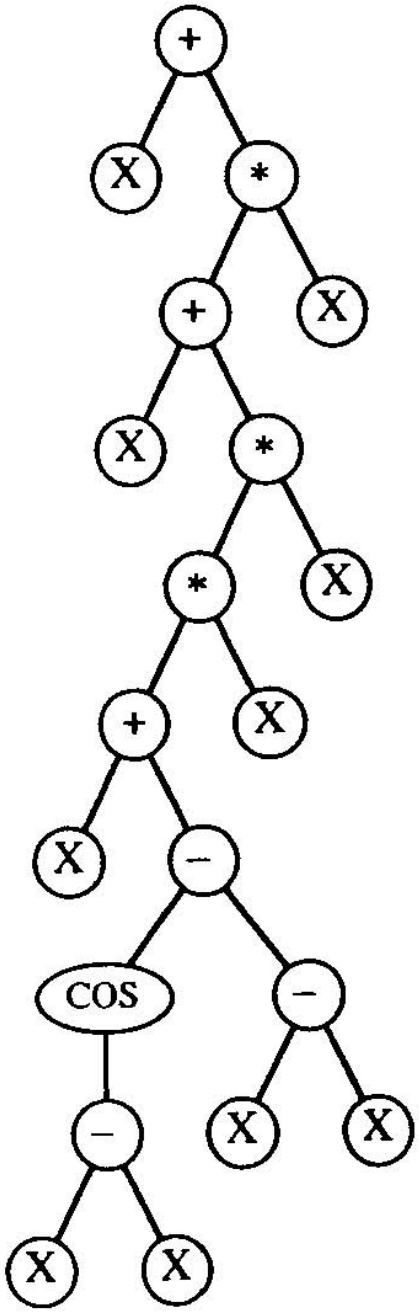
Fitness Curves for the simple symbolic regression problem

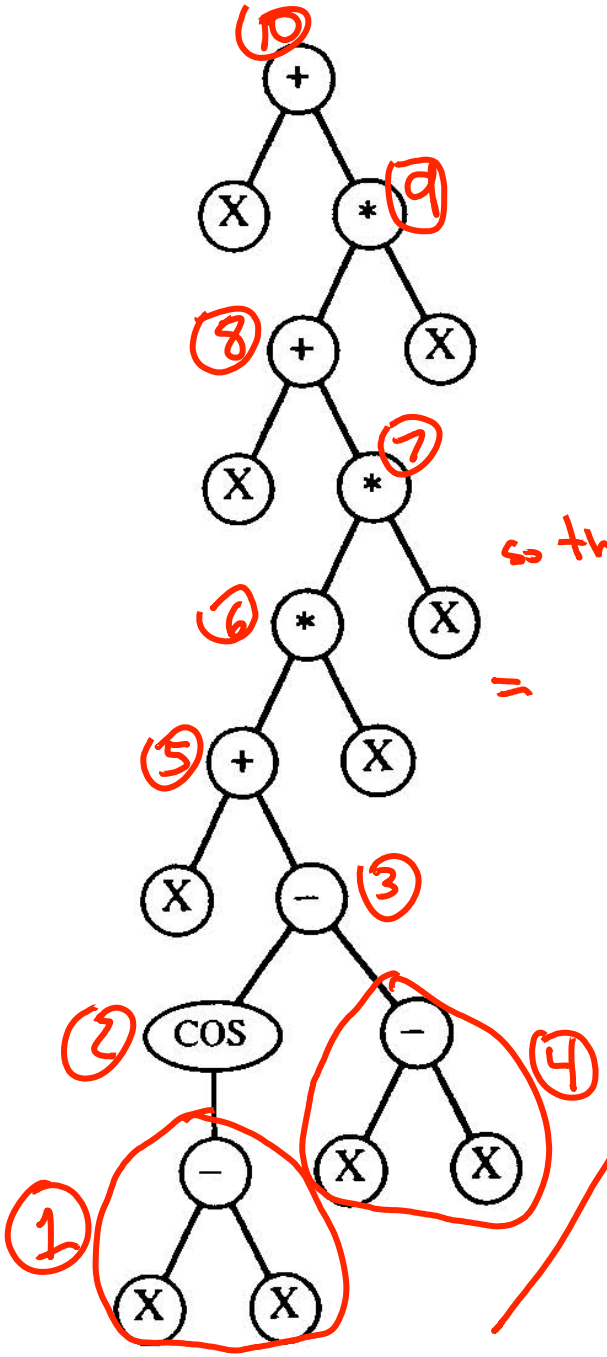


Usually we only plot the bottom line, which looks flat because of the scale. The bad solutions are REALLY bad.

Best Solution from Generation 34

What does this equal??? Are there unnecessary terms?





bottom lim become

$x(x(x+\underbrace{1}_{\cos(0)}-0)) = x^3+x^2$

so then $x+(x(x+(x^3+x^2)))$

=

Best Solution from Generation 34

$(x+(x+(x+(x+(x+(x+(x+(x-x))))))))$

$\underbrace{\cos(x-x)}_{=1}$ $\underbrace{(x-x)}_{=0}$

Video of Applications

- Econometrics—forecasting prices as a function of gross national product and money supply. Fit equation to the past and then look at its ability to forecast.

- In this example the regression includes two independent variables.

Getting Constants into equation

- X+x/x = 2 is one way to introduce constants
- Also can put numbers in terminals

Best method for the problem

- If you know that the best function is a polynomial, would genetic programming be the best method to solve this problem?
- Why or Why not?
- When would you recommend using Genetic Programming for Symbolic Regression?

Genetic Programming: Symbolic Regression

- In the previous slides, we discussed use of genetic programming for "symbolic regression".
- Recall that for regular regression (or spline fits), the **form** (e.g. polynomial) of the terms in the function is fixed.
- In polynomial regression the coefficients of the terms are selected to fit the data (and some coefficients may be zero or some terms eliminated because they do not add enough to the quality of the fit).
- handout 11-3-10

Genetic Programming: Symbolic Regression

- The difference with symbolic regression with genetic programming is that the

Mathematical form as well as the coefficients are selected by the algorithm

since your decision variable (which we will call an "**S-expression**") is composed of different algebraic expressions (e.g. including terms like x/y, cos x, or exp(x-y/z)).

What possible advantage might that approach have?

Application to Econometrics

- Here is an application of symbolic regression to econometric data.
- Recall that the symbolic regression can be used when you have more than one regression variable.

- Video
- video

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.

Figure Artificial Intelligence Problem

- 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),

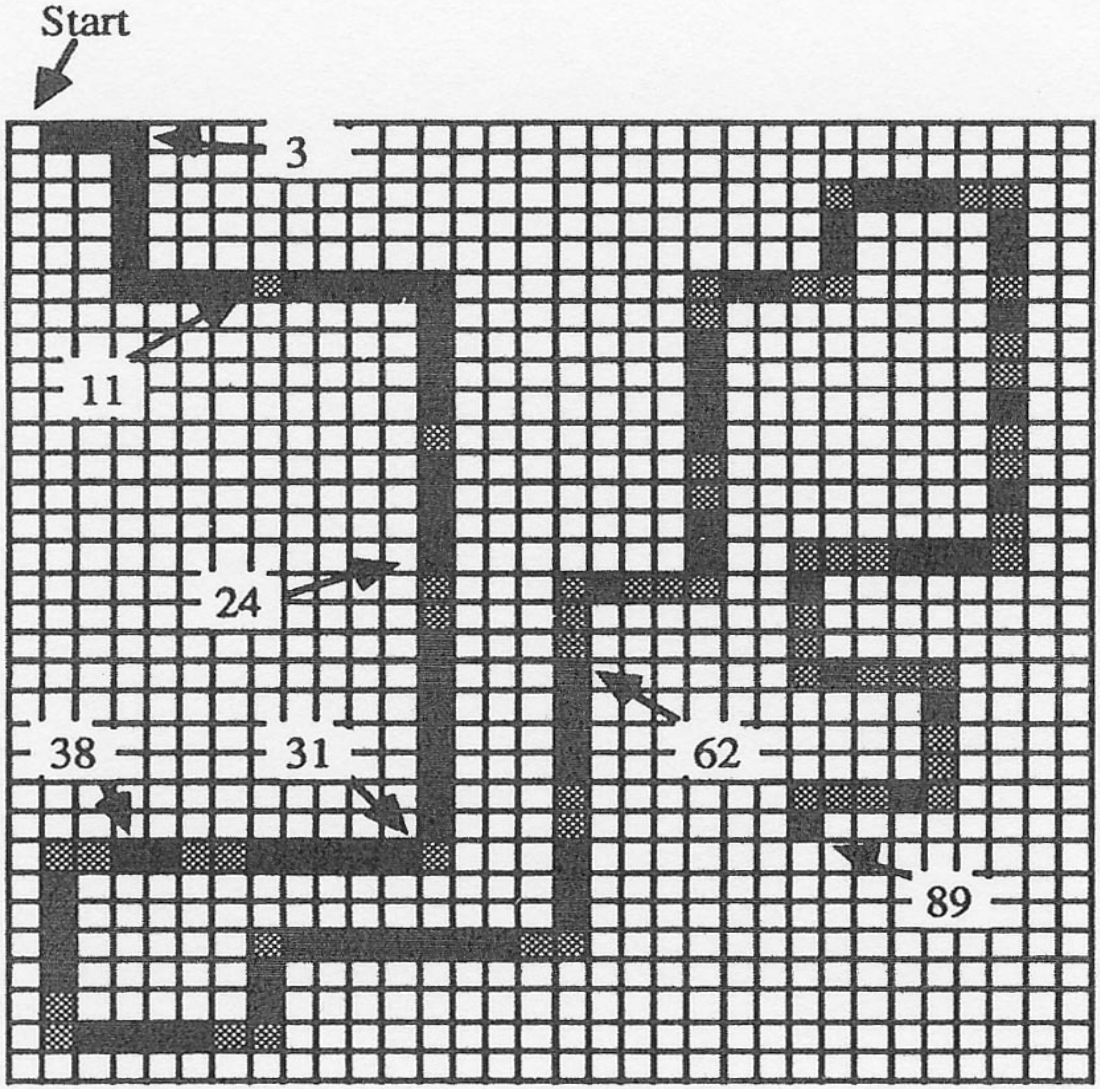
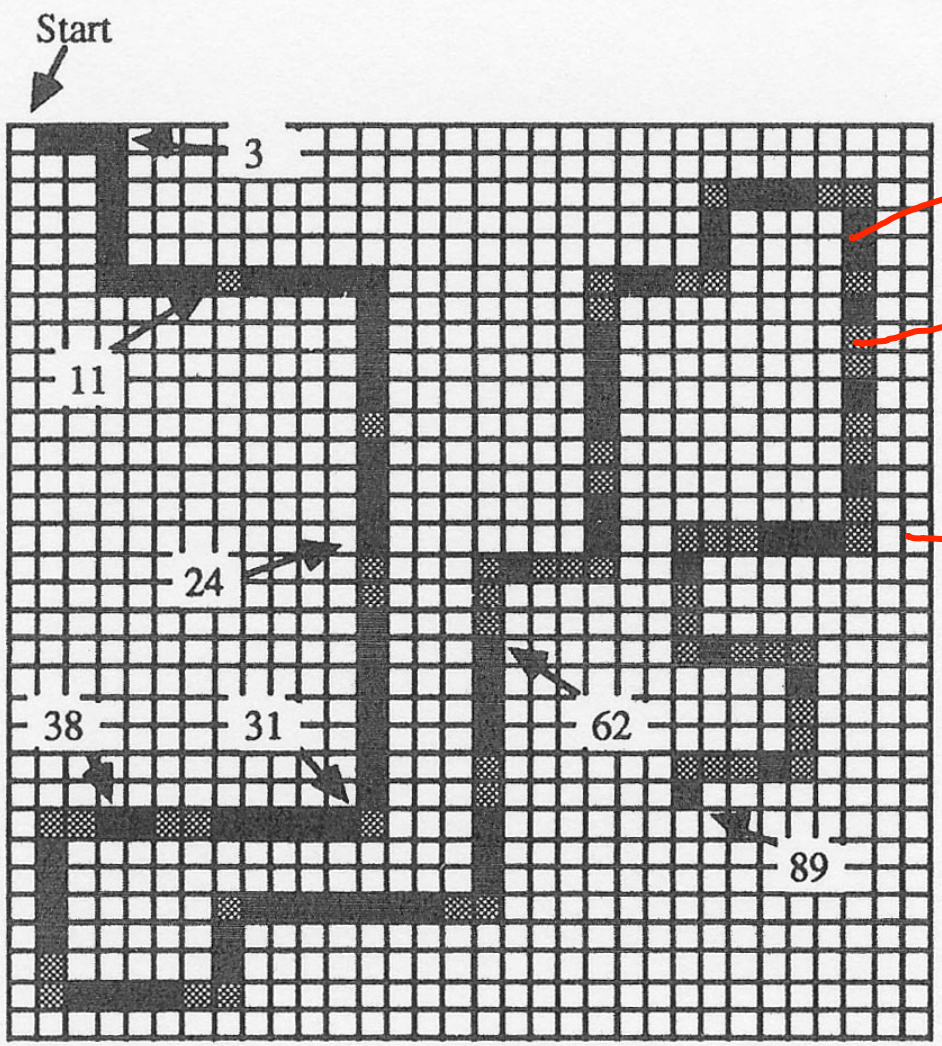


Figure 3.6 The Santa Fe trail for the artificial ant problem.

Fig. 3.6 Santa Fe Trail (from Koza book, 1992)



black = food

checkered = no food and on trail

blank = no food

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}**

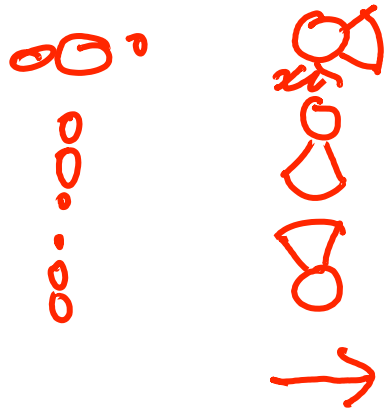
b. It can move straight ahead (which counts as 1 time period). If it moves into a square with food, it eats the food.

Terminal for this is **{move}**

Board Notation for Ant moves

Top

cartoon



look ahead

look right

look left

move in direction looking

- 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 arg 1, 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** arg 1 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}

BOARD Functions

- The **function PROGNm arg 1, arg2,…,argm** first implements arg1, then implements arg2, and so forth until it lastly implements argm. The argi's can be the terminals, right, left or move or other functions.
- The function **If food ahead** arg 1 arg2

will do arg1 if the ant senses that there is food straight ahead and will do arg2 if there is no food straight ahead.

- We let the **function set** for this problem be
- {If-food-ahead, PROGN2, PROGN3}

e.g. PROGN2 (RIGHT, Left) ← goes back to original position

- Some examples:
- **PROGN2 (RIGHT) (LEFT)** does what? Will it be successful at getting reaching the goal?
-
- What about **IF-FOOD-AHEAD (RIGHT) (MOVE)**? How much food will it get? Why?

- How much food would **IF-FOOD-AHEAD (MOVE) (MOVE)** obtain?

BOARD

- Some examples:
- **PROGN2 (RIGHT) (LEFT)** does what? Will it be successful at getting reaching the goal?
- stays same place
- What about **IF-FOOD-AHEAD (RIGHT) (MOVE)**? How much food will it get? Why?

avoids food - won't get any

- How much food would **IF-FOOD-AHEAD (MOVE) (MOVE)** obtain?

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.
- 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.

Objective:	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.
Terminal set:	`(LEFT), (RIGHT), (MOVE).`
Function set:	`IF-FOOD-AHEAD, PROGN2, PROGN3.`
Fitness cases:	One fitness case.
Raw fitness:	Number of pieces of food picked up before the ant times out with 400 operations.
Standardized fitness:	Total number of pieces of food (i.e., 89) minus raw fitness.
Hits:	Same as raw fitness for this problem.
Wrapper:	None.
Parameters:	$M = 500$. $G = 51$.
Success predicate:	An S-expression scores 89 hits.

End of lecture 11-18-11

BOARD

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

Objective:	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.
Terminal set:	`(LEFT)`, `(RIGHT)`, `(MOVE)`.
Function set:	`IF-FOOD-AHEAD`, `PROGN2`, `PROGN3`.
Fitness cases:	One fitness case.
Raw fitness:	Number of pieces of food picked up before the ant times out with 400 operations.
Standardized fitness:	Total number of pieces of food (i.e., 89) minus raw fitness.
Hits:	Same as raw fitness for this problem.
Wrapper:	None.
Parameters:	$M = 500$. $G = 51$. = how many program evaluations?
Success predicate:	An S-expression scores 89 hits.

figure 7.12 EXAMPLES OF POSSIBLE S-EXPRESSIONS (VALUES OF DECISION VARIABLES)

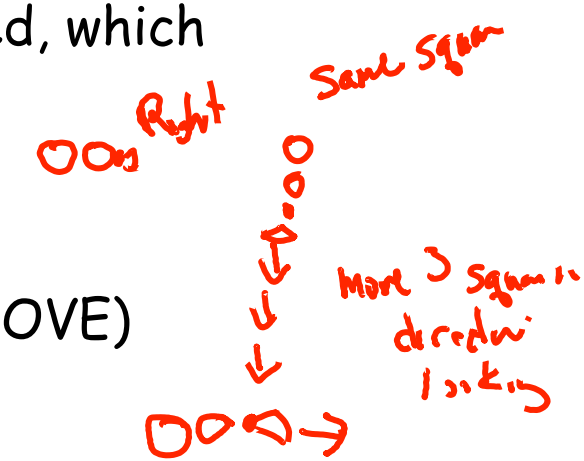
- Figure 7.12 is one of the solutions obtained, which Koza calles the "quilter". Its program is

- PROGN3 (RIGHT)
- (PROGN3 (MOVE) (MOVE)(MOVE)
- (PROGN2 (LEFT) (MOVE)))
- You should understand this and be able to indicate how the ant will move given an expression like this.

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

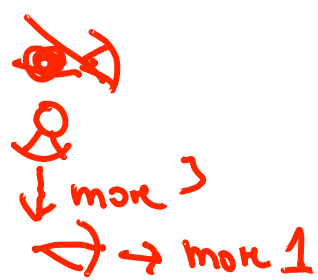
BOARD: figure 7.12 EXAMPLES OF POSSIBLE S-EXPRESSIONS (VALUES OF DECISION VARIABLES)

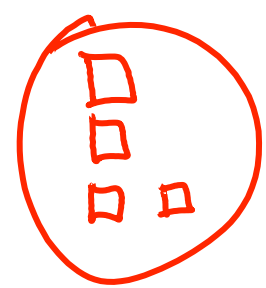
- Figure 7.12 is one of the solutions obtained, which Koza calles the "quilter". Its program is



- PROGN3 (RIGHT)
- (PROGN3 (MOVE) (MOVE)(MOVE)
- (PROGN2 (LEFT) (MOVE)))

so





Path of Quilter

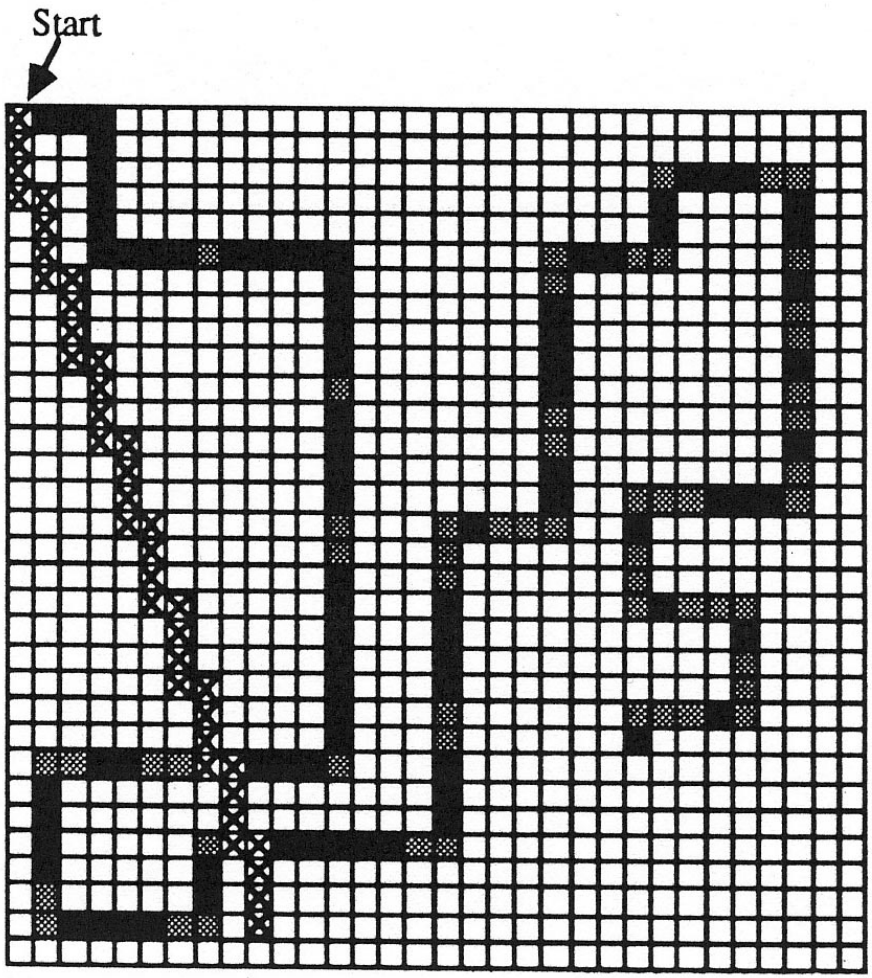


Figure 7.12 Path of the quilter from generation 0 of the artificial ant problem.

Path of the Looper

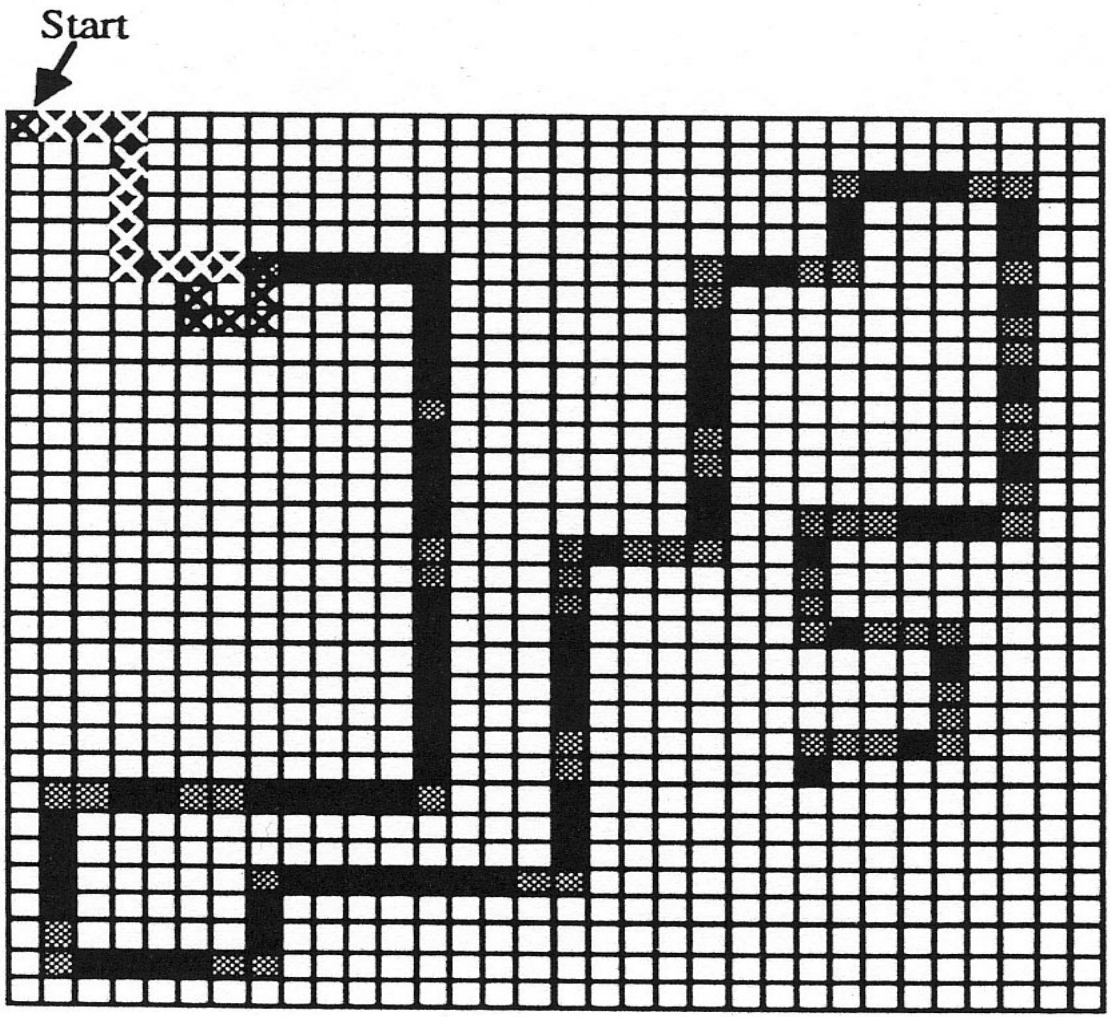


Figure 7.13 Path of the looper from generation 0 of the artificial ant problem.

figure

- 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

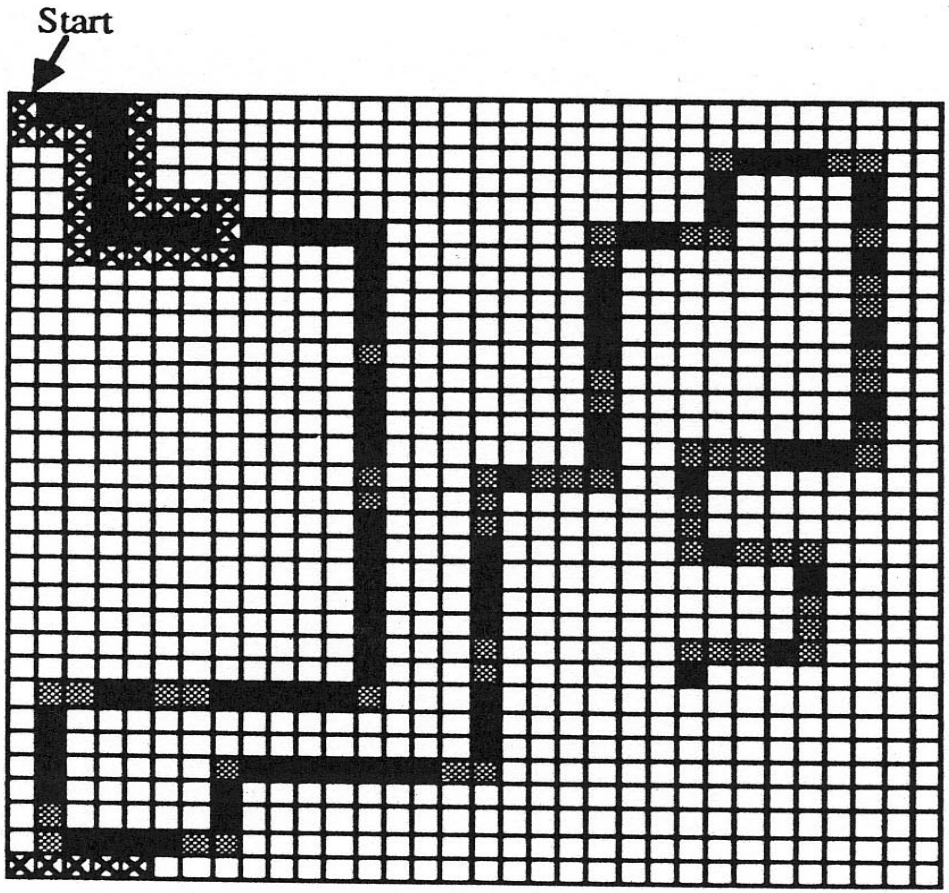


Figure 7.14 Path of the avoider from generation 0 of the artificial ant problem.

figure

Figure 7.14 shows the “avoider”

```
(           IF-FOOD-AHEAD          (RIGHT)
                                   (IF FOOD-AHEAD          (RIGHT)
                                                           (PROGN2 (MOVE) (LEFT
```

Not handout Best of Run Individual for Artificial Ant

- So this algorithm is:
- Move ahead if there is food
- If no food turn left and move ahead if food
- Otherwise turn right twice and move ahead if food
- Otherwise turn left (so you are facing in original direction) and move even though there is no food.

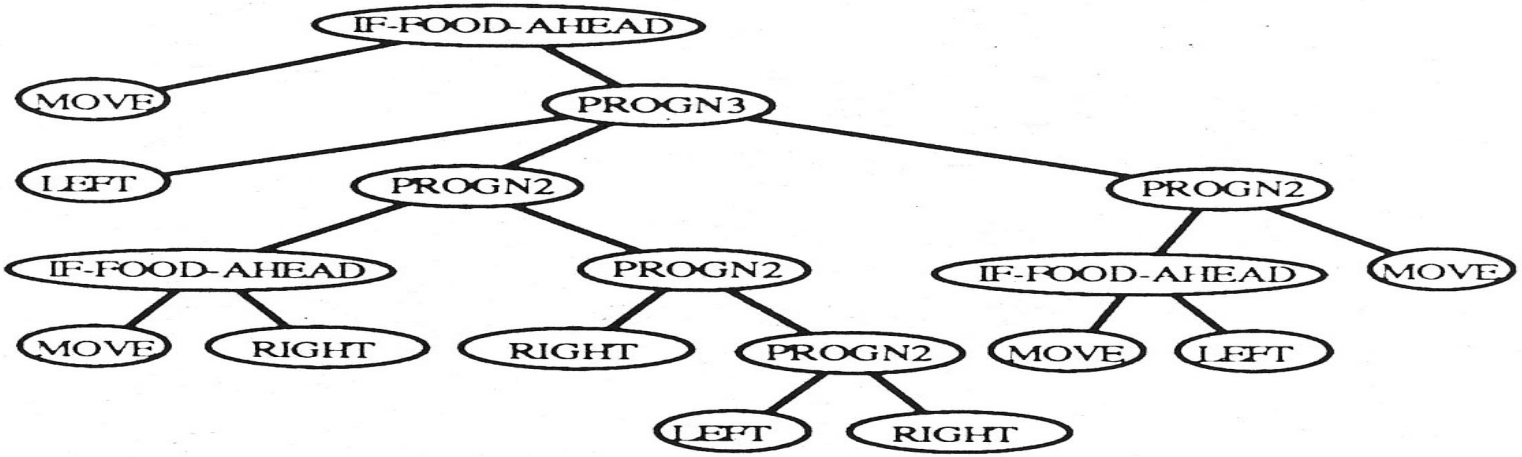


Figure 7.16 Best-of-run individual for the artificial ant problem.

Best of Run Individual for Artificial Ant

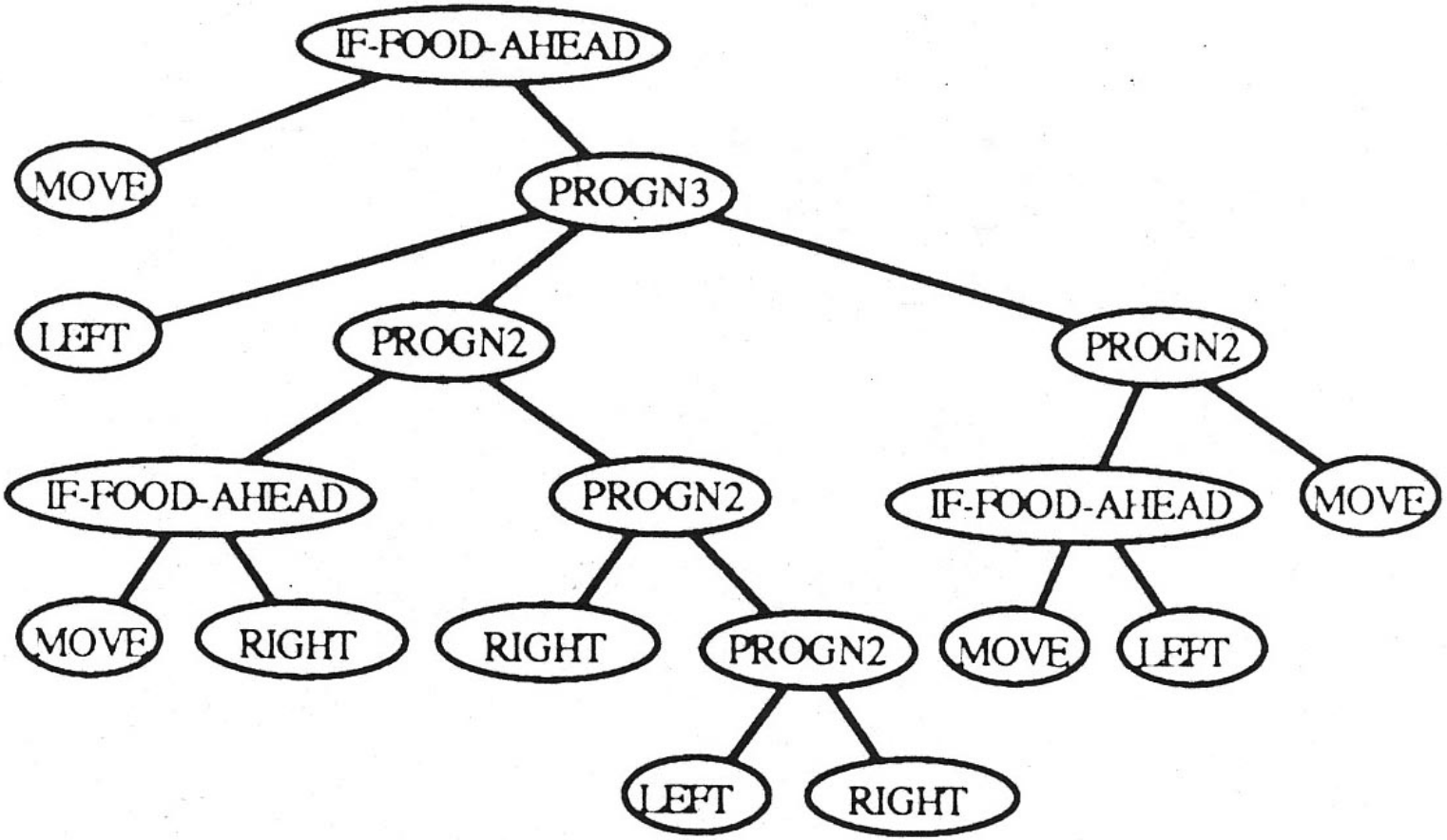


Figure 7.16 Best-of-run individual for the artificial ant problem.

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.)

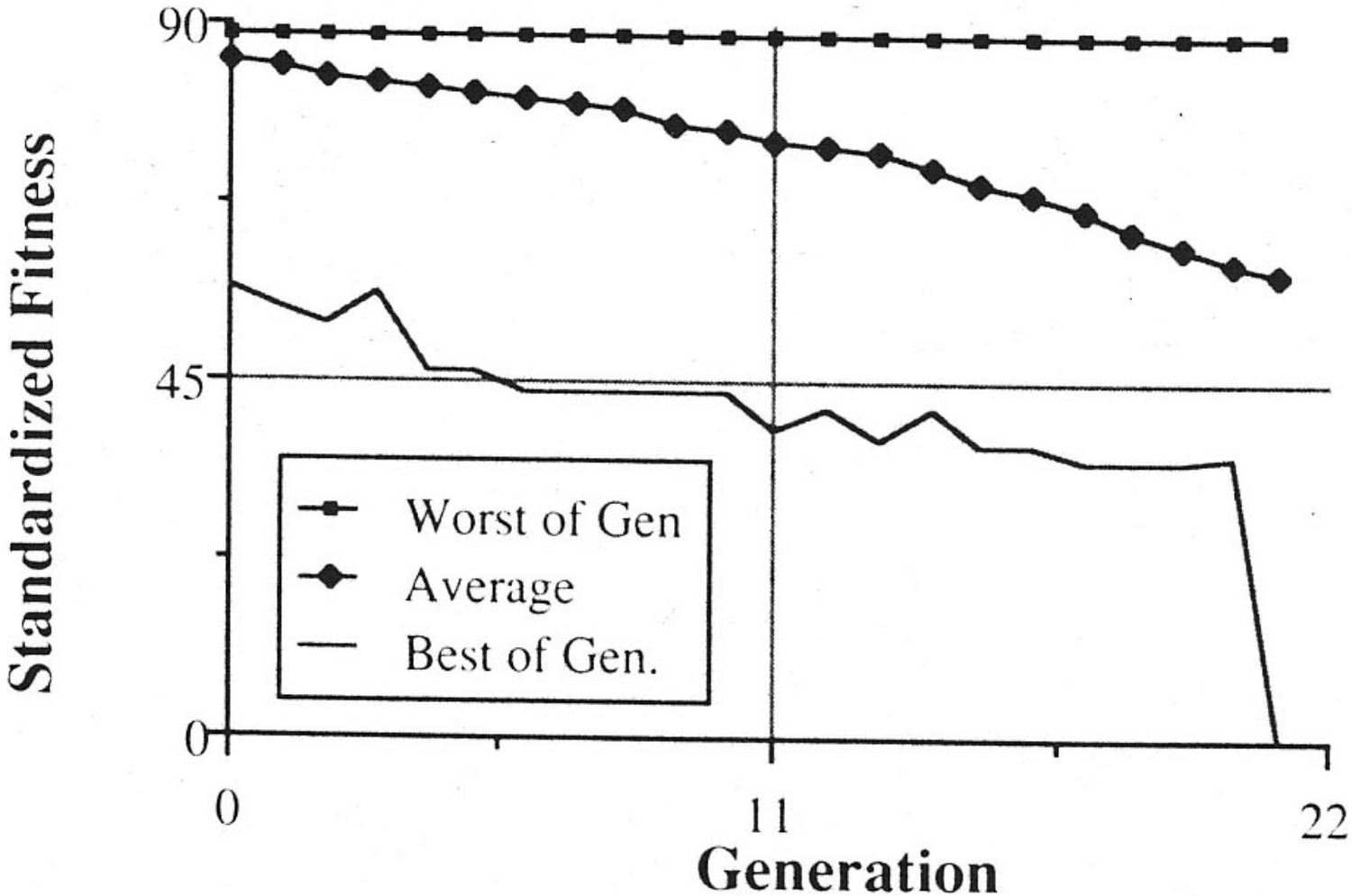


Figure 7.15 Fitness curves for the artificial ant problem.

Best of Run Individual for Artificial Ant

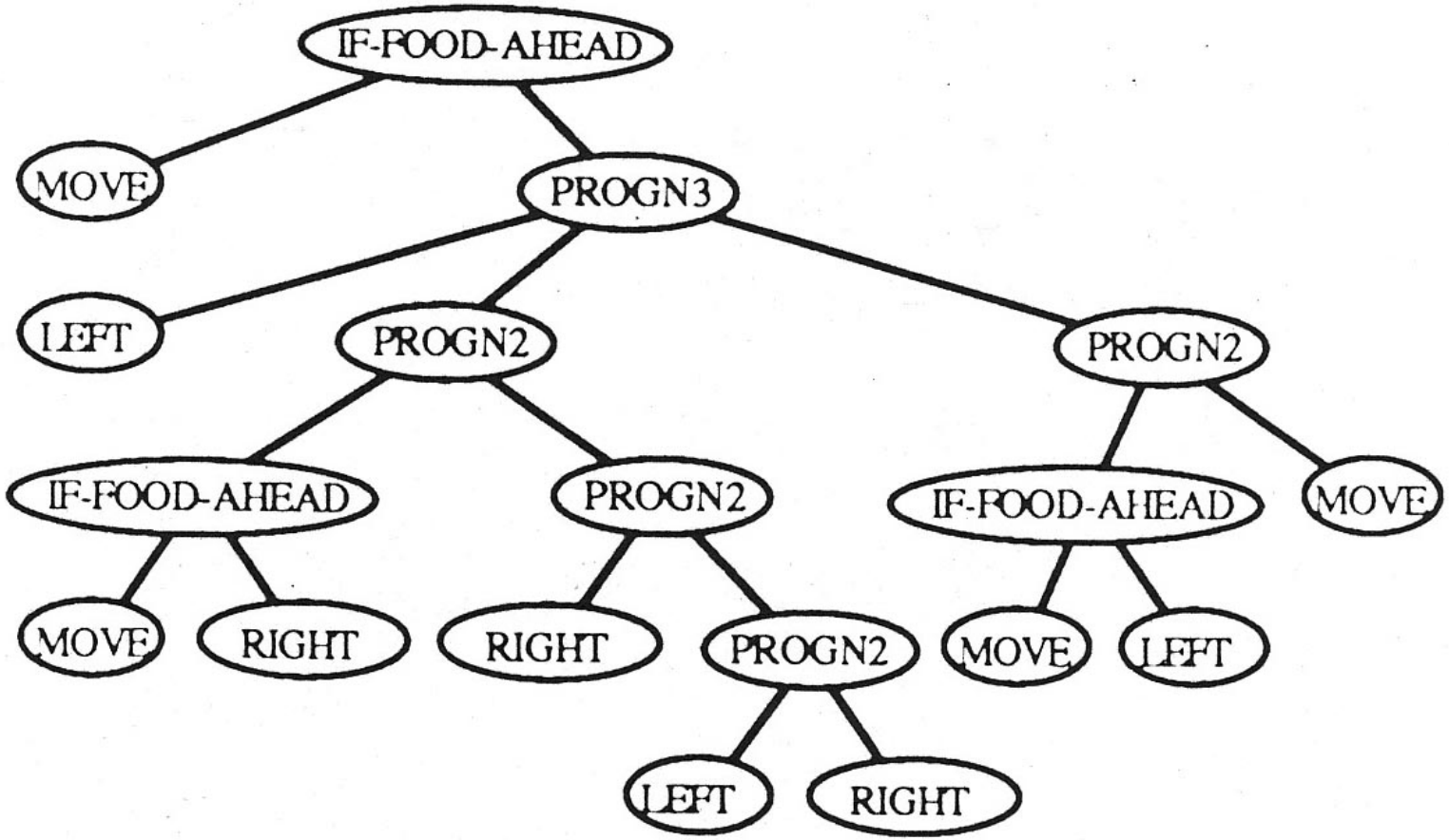


Figure 7.16 Best-of-run individual for the artificial ant problem.

Figure 7.16

- 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-exprssion senses no food, the ant enters the three step PROGN3 sequence immediately below the IF-FOOD-AHEAD test.

Figure 7.16

- 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, tthe 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?

Handout 11-5-10 genetic programming

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?)

BOARD:

- If you have all the characteristcis listed as for Sante Fe, then the program is optimial.
- If you have a simpler problem, you could make it faster by dropping some things.
- For example what if you had a straight line with only single food gaps and no corners , you could just use "Move" and that would be the fastest-no looking right or left.

Power of GP for Artifical 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 encouters 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?

Not handout 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? Not necessarily
- What if you only two steps to the first corner Would your program still be able to work? yes
- What if the first corner goes left instead of right? OK
- What if the first food gap is replaced with food? OK
- What if you had two empty boxes before turning a corner? No
- 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?

Stopped here 11-21 but did discuss Los Altos graph and the empty boxes after the corner.

Review

- Genetic programming can be used to find a set of instructions and create a computer code to guide an "artifical ant" to search through an area to find food.
- The genetic programming algorithm is similar to GA with crossover and fitness based selection, but the decision variable is a parse tree.
- The optimal solution for the "Sante Fe" is shown on the next slide.

Review: Best of Run Individual for Artificial Ant on Sante Fe Trail

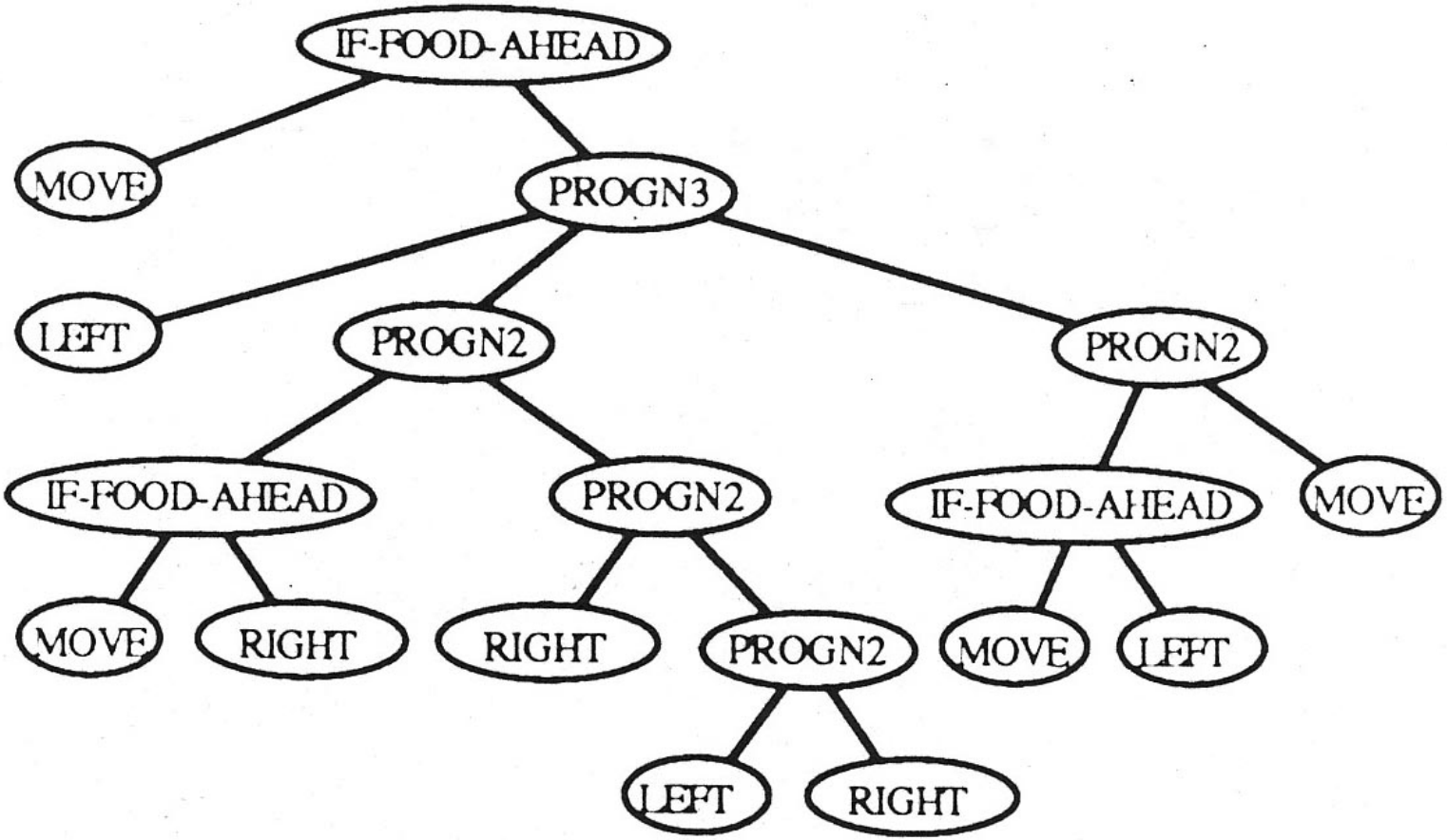


Figure 7.16 Best-of-run individual for the artificial ant problem.

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, PlanataryExploration.
- 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 Autonmous Vehicles (AV)?

- We do not want to program an AV just for one specific course (since there are easier was 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 <u>based on what it sees or otherwise senses.</u>
- Most AV have cameras and/or other sensing devices.

Los Altos HillsTrail

O = no food on trail

Los Altos Hill

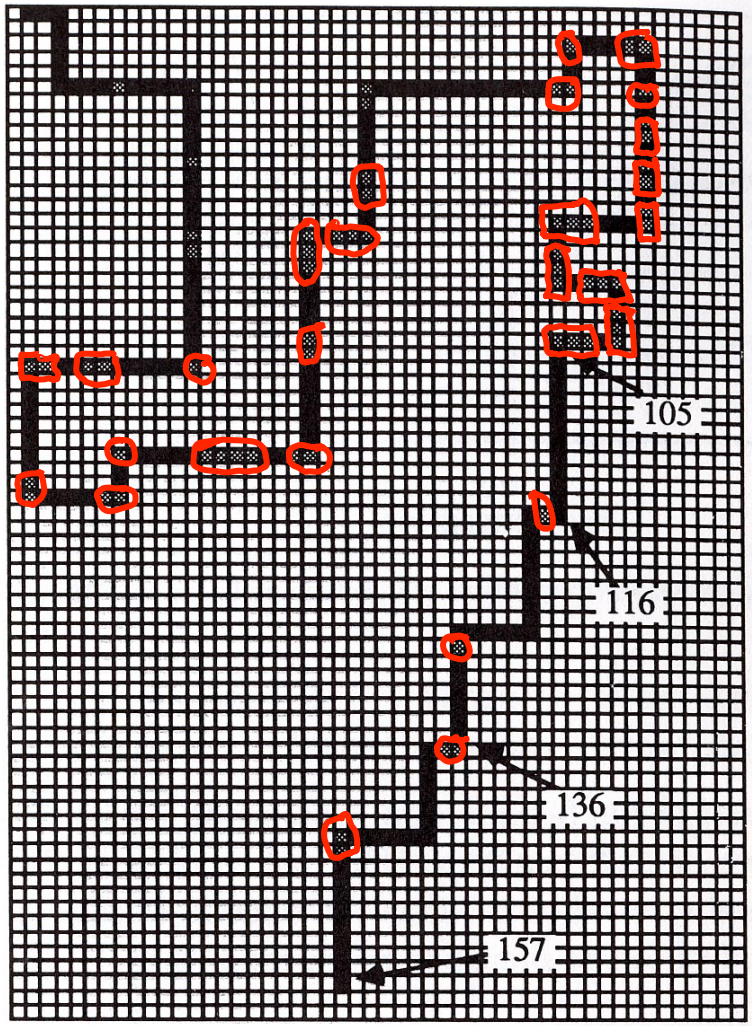


Figure 7.17 The Los Altos Hills trail for the artificial ant.

Are any of these gaps going to cause the Ant to go in the wrong direction with the previous program?

boardLos Altos HillsTrail

O = no food on trail

Los Altos Hill

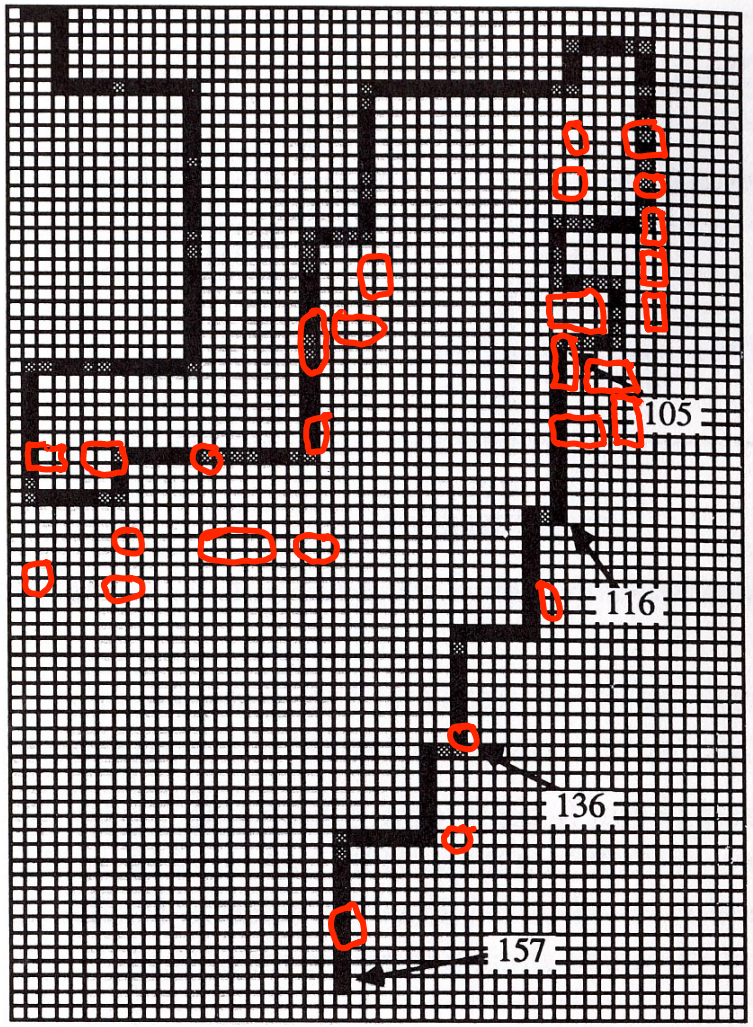


Figure 7.17 The Los Altos Hills trail for the artificial ant.

Are any of these gaps going to cause the Ant to go in the wrong direction with the previous program?

136 is new since there is food gap AFTER the corner

Los Altos Trail

Los Altos Trail

- What is different about this trail from Sante Fe trail?
- Will the old program work? Why or Why Not?
- A later graph will show the number of "hits" which is the amount of food gathered.

```
(PROGN4 (IF-FOOD-AHEAD (PROGN2 (PROGN3 (MOVE) (PROGN2
(MOVE) (MOVE)) (RIGHT)) (IF-FOOD-AHEAD (MOVE)
(IF-FOOD-AHEAD (IF-FOOD-AHEAD (LEFT) (LEFT))
(PROGN4 (PROGN2 (IF-FOOD-AHEAD (MOVE) (RIGHT))
(MOVE)) (RIGHT) (MOVE) (MOVE))))) (PROGN4 (PROGN2
(IF-FOOD-AHEAD (MOVE) (RIGHT)) (MOVE)) (RIGHT)
(MOVE) (MOVE))) (IF-FOOD-AHEAD (IF-FOOD-AHEAD (MOVE)
(IF-FOOD-AHEAD (IF-FOOD-AHEAD (MOVE) (LEFT))
(IF-FOOD-AHEAD (LEFT) (RIGHT)))) (IF-FOOD-AHEAD
(LEFT) (RIGHT))) (PROGN2 (PROGN3 (MOVE) (MOVE)
(RIGHT)) (IF-FOOD-AHEAD (PROGN2 (PROGN3 (MOVE)
(PROGN2 (MOVE) (MOVE)) (RIGHT)) (IF-FOOD-AHEAD
(IF-FOOD-AHEAD (MOVE) (MOVE)) (MOVE))) (MOVE))) (MOVE)).
```

The above best-of-run S-expression contains 66 points. Not surprisingly, solving the more difficult Los Altos Hills trail required an S-expression with more internal and external points than the solution for the original Santa Fe trail. This individual can be simplified to the following:

```
(PROGN7 (IF-FOOD-AHEAD
          (PROGN5 (MOVE) (MOVE) (MOVE) (RIGHT)
                  (IF-FOOD-AHEAD
                    (MOVE)
                    (PROGN5 (RIGHT) (MOVE)
                            (RIGHT)
                            (MOVE) (MOVE))))
          (PROGN5 (RIGHT) (MOVE) (RIGHT)
                  (MOVE) (MOVE)))
        (IF-FOOD-AHEAD (MOVE) (RIGHT))
        (MOVE)
        (MOVE)
        (RIGHT)
        (IF-FOOD-AHEAD
          (PROGN5 (MOVE) (MOVE) (MOVE) (RIGHT)
                  (MOVE))
          (MOVE))
        (MOVE)).
```

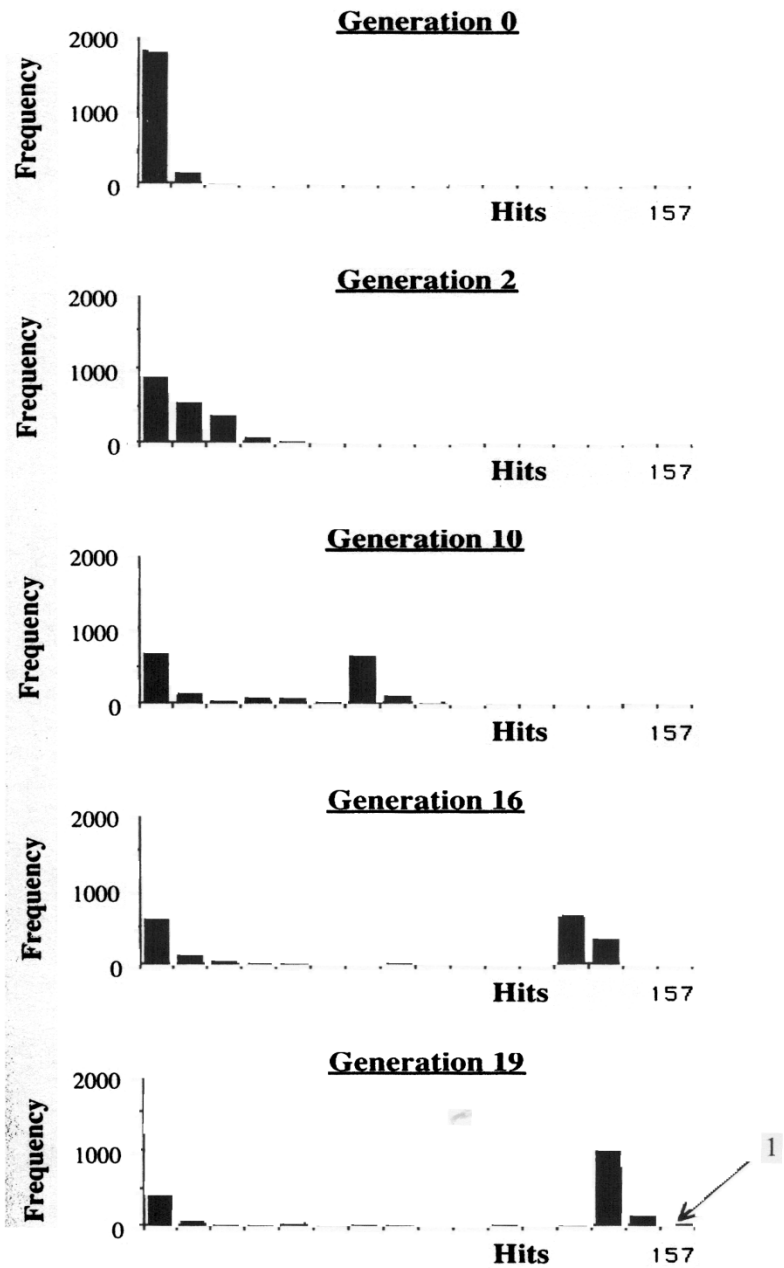


Figure 7.21 Hits histograms for generations 0, 2, 10, 16, and 19 for artificial ant problem with the Los Altos Hills trail.

End 11-21 handout?

Computational Issues: How Many Program Evaluations?

- Maximum number of program evaluations is _____=_____ for the ant problem.

- If each program took 1 minute (or hour) to evaluate, how long does it take to evaluate the maximum number of iterations in serial?

- Could you do this in parallel? What is the maximum number of processors you could use?

BOARD: How Many Program Evaluations

- Maximum number of program evaluations is
- M G _____ = _____ 25,500 for the ant problem.
- If each program took 1 minute to evaluate, how long does it take to evaluate the maximum number of iterations in serial? _____ hour
- Could you do this in parallel? What is the maximum number of processors you could use?

= 25,500 hours ≅ 1062 days ≅ 3 years

- Which with 500 processor at best is 3/500=52 hours.
- If program time is 1 minute then 25500 minutes= 17 days. The best you could do with 500 processors is 17day/500= about 1 hour. With 500 processors.

Yes, 500 processors

Even if each program take 1 minute time required in serial ~ almost 2 months

Computational Conclusions

- To solve even some relatively simple problems is computationally expensive.
- The serial wall clock time is prohibitive for simulation (objective function evaluations times) that are modest, e.g. a minute.
- The process is highly parallelizable, but it is costly to run a large number of processors (since computer cost is related to the number of processors being used)

Computational Issues

- However, as computing becomes less expensive and availability of many processor computers increases, the feasibility of this method will also increase.

Structure Complexity

Structural complexity= total number of terminals and functions in a program

- Graph shows the average and the best
- What can you conclude from this?

Board: Structure Complexity

Structural complexity= total number of terminals and functions in a program

- Graph shows the average and the best

- What can you conclude from this?
- Initially you generate random population, but don't start with really complex programs.
- As it evolves you do get some complex programs.
- However, eventually eliminate the overly complex programs (since some of the simpler programs you discover are better).

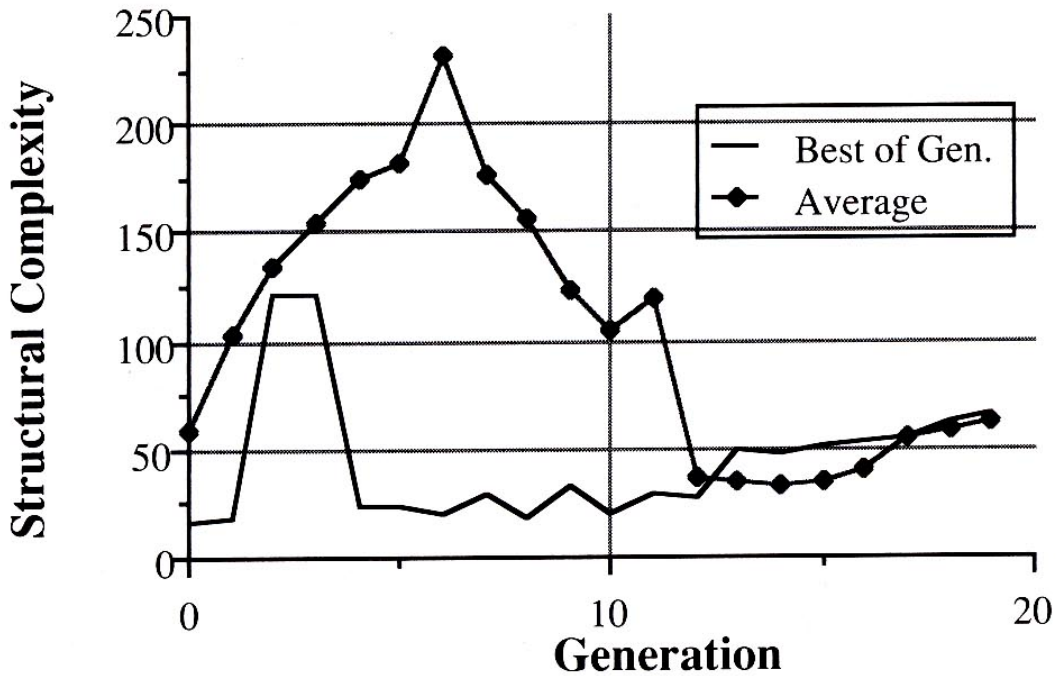


Figure 7.19 Structural complexity curves for the artificial ant problem with the Los Altos Hills trail.

Fitness curves for Los Altos Hills Trail

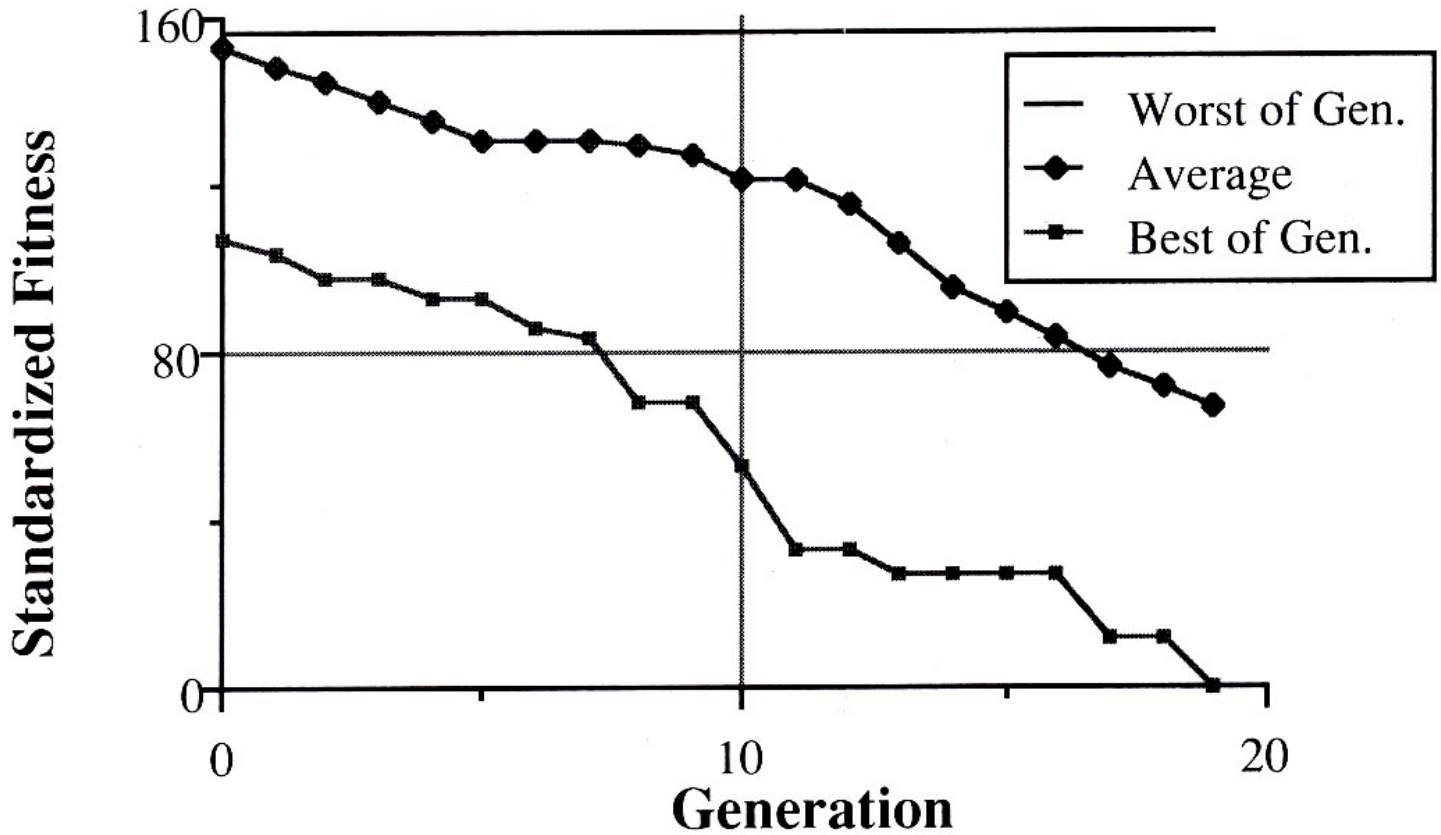


Figure 7.18 Fitness curves for the artificial ant problem with the Los Altos Hills trail.

Number of Terminals

- What if you wanted to negotiate the Sante Fe trail with a program with fewer terminals?
- Could you eliminate the terminal "Right"?
- What would be the effect on the length of time in the (true) optimal solution?
- What would be the effect of the number of generations it takes to reach a good solution?

Solution Obtained in 19 generations for Sante Fe Trail without a “Left” Terminal

```
(PROGN2 (PROGN2 (IF-FOOD-AHEAD (MOVE) (RIGHT))
                (PROGN2 (MOVE) (RIGHT)))
        (PROGN2 (IF-FOOD-AHEAD
                  (PROGN3 (MOVE)
                          (PROGN2 (MOVE) (RIGHT))
                          (RIGHT))
                  (RIGHT))
                (IF-FOOD-AHEAD (RIGHT)
                               (RIGHT)))).
```

Interpretation

- The Selection of Terminals will affect the solution that you obtain.
  - In this case the solution on previous slide will also pick up the food, but it will take more time. (Why?)
  - However the time alloted (400 time steps) is more than is needed so even this solution without a Left can pick up all the food.
- Solution time can also be faster if you have fewer terminals, assuming it is possible to get a good solution satisfying the constraints.

Conclusions on Genetic Programming

- This is a very inventive application of the ideas of Genetic algorithms to decisions that are "programs"
- A program is not only a code, but it is also the instructions/equations in the code.
- Genetic program is hence inventive because it is helping to identify the set of instructions or equations that will solve a problem

Conclusions on Genetic Programming (cont.)

- We looked at two different kinds of GP applications:
  - Symbolic regression-important to identify equations and their coefficients when you don't know the form (e.g. linear, fractional, etc.) of the equation
  - Artificial Intelligence for Autonomous vehicles—a very different application that the ones we have previously discussed

Conclusions on Genetic Programming (cont)

- Computational Effort to generate even a simple symbolic regression or autonomous vehicle program can be significant.
  - Very useful outcome and for small problems the computation is worth the effort

- However, both the computational effort and the search mechanism for a solution may not be practical for complex programs.
  - Perhaps by breaking up a complex program into smaller programs could make GP a more practical tool.

Conclusions on GP (continued)

- Given the increasing amount of cheap computing power, the usefulness of Genetic Programming is expected to grow.
- There are of course ways to improve the algorithm
  - For example Prof. Lipson's lab has produced a GP symbolic regression code that uses multi-optimization to improve the quality of the solution.

Other slides

- P. 594 and and Fextra p
- Fig 2410

- Defining prob of success fig 24.12
- Fig. 25.1 etc. Need def

- Encapsulation—interesting concept—what does it mean?

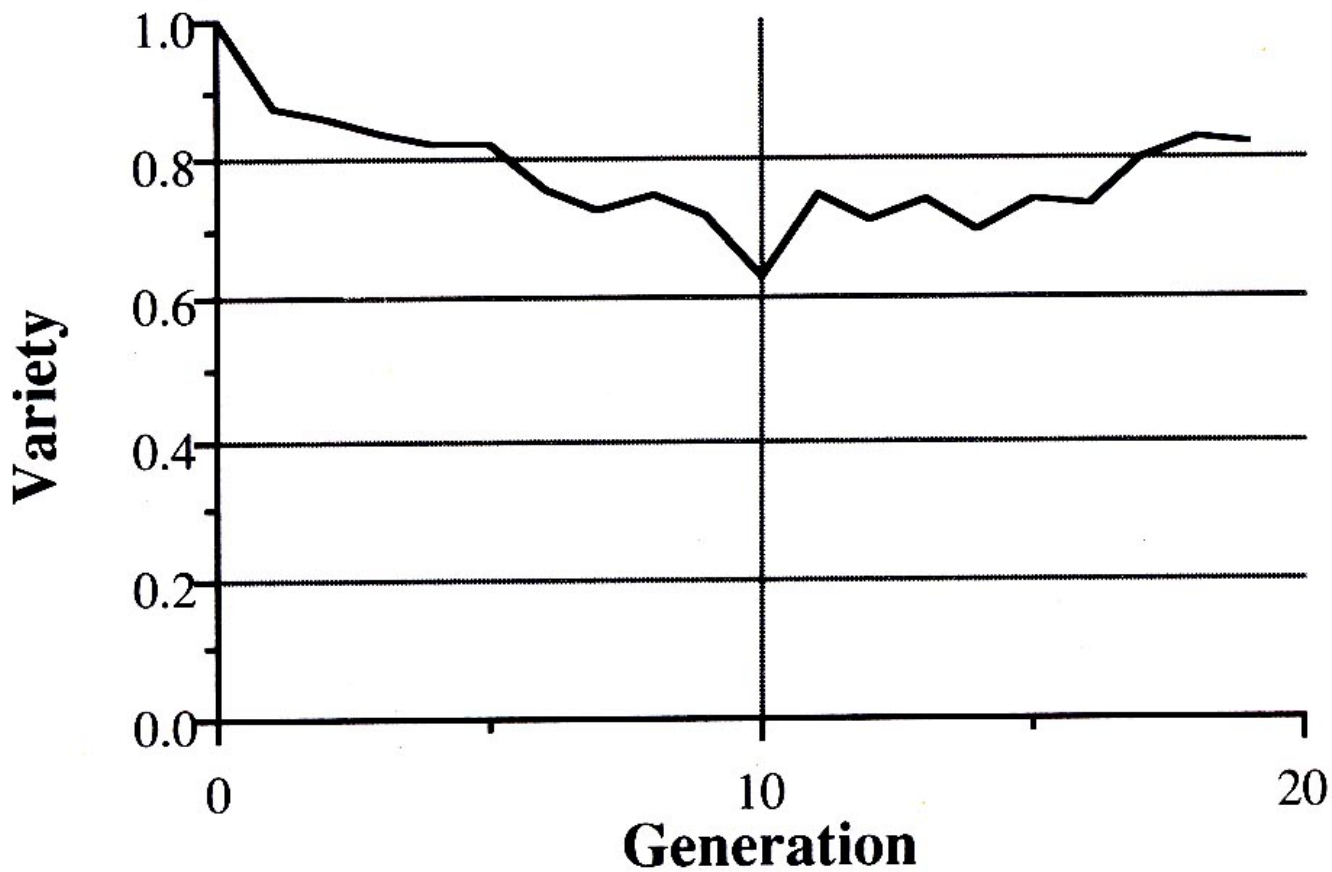


Figure 7.20 Variety curve for the artificial ant with the Los Altos Hills trail.

F_{extra} = {IF-FOOD-AHEAD, PROGN2, PROGN3, +, -, *, %, SIN, COS, EXP, RLOG}.

The terminal set for the symbolic regression problem consisted of the independent variable X:

T = {X}.

The terminal set of the symbolic regression problem is similarly enlarged so that it becomes

T_{extra} = {X, (MOVE), (RIGHT), (LEFT)}.

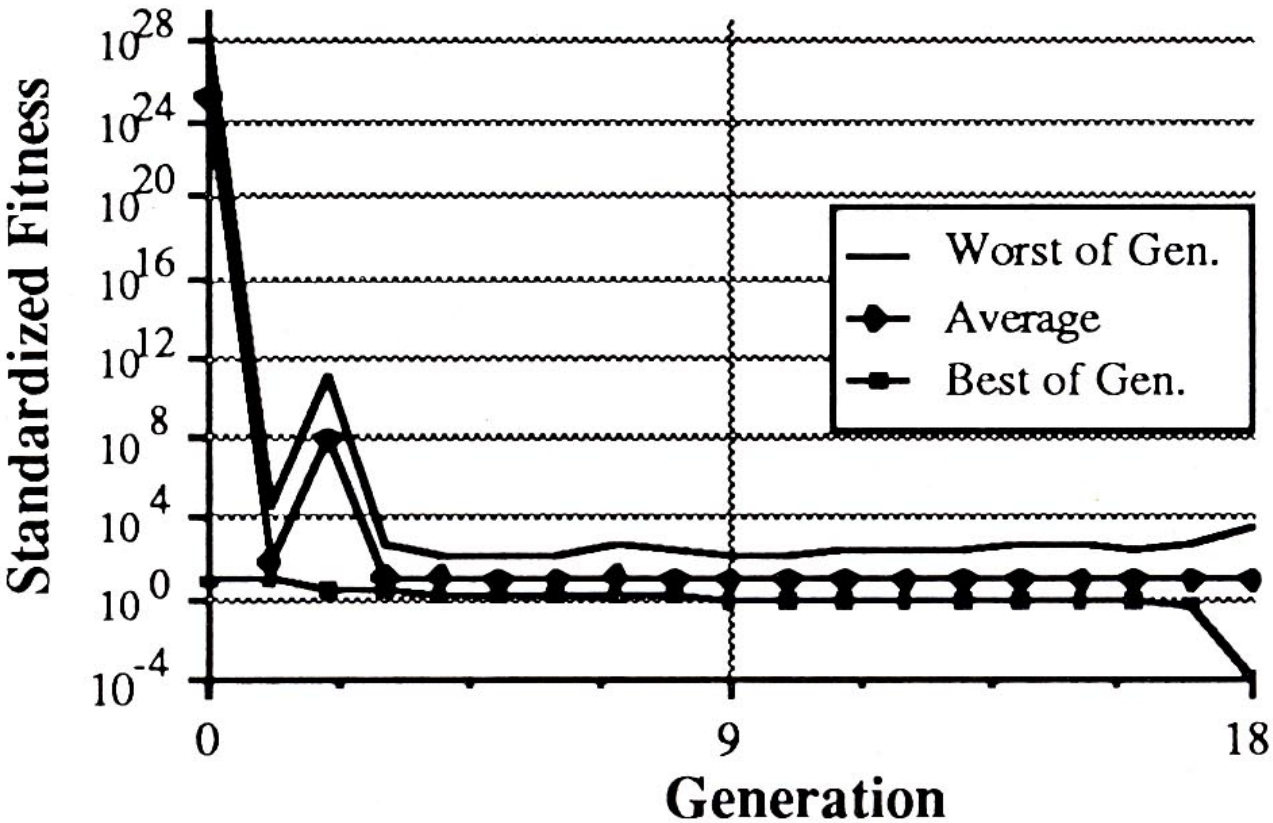


Figure 24.9 Fitness curves for the simple symbolic regression problem using T_{extra} and F_{extra}.

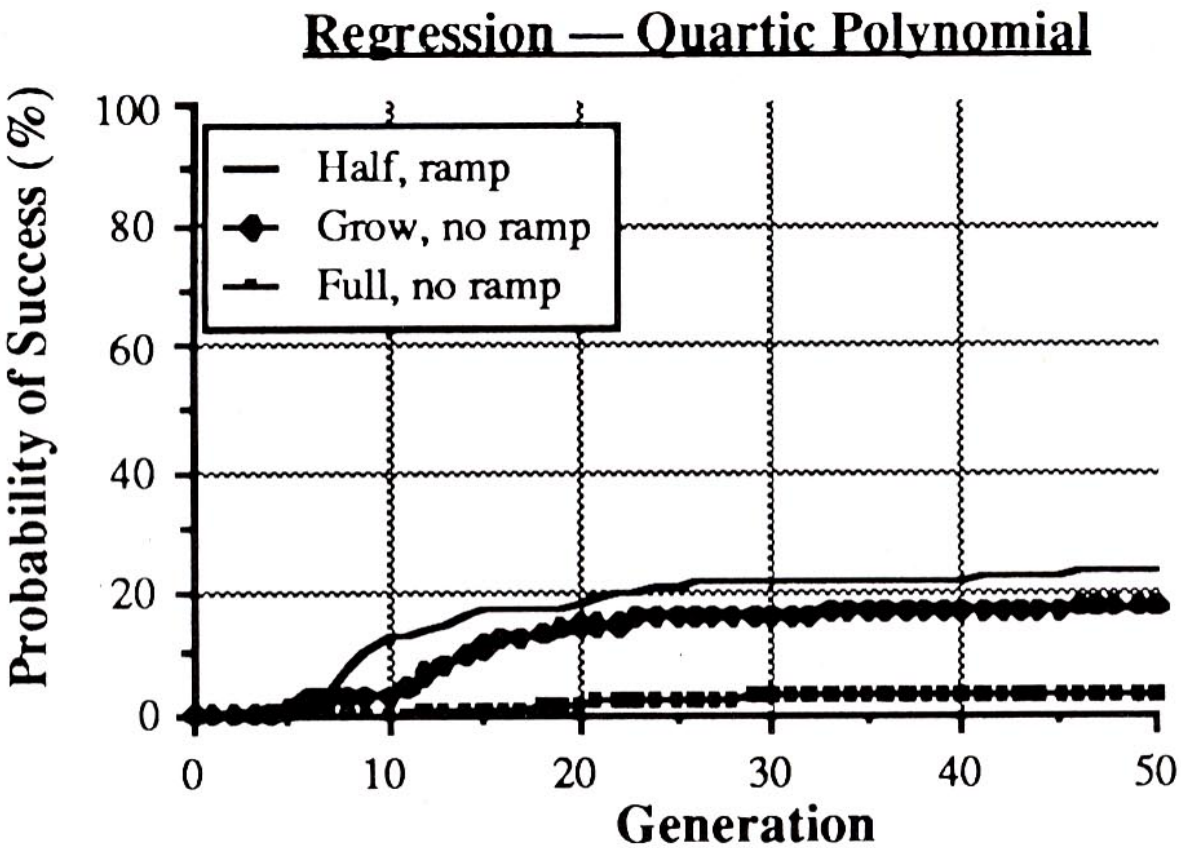


Figure 25.1 Comparison of three methods of creating the initial random population for the simple symbolic regression problem.

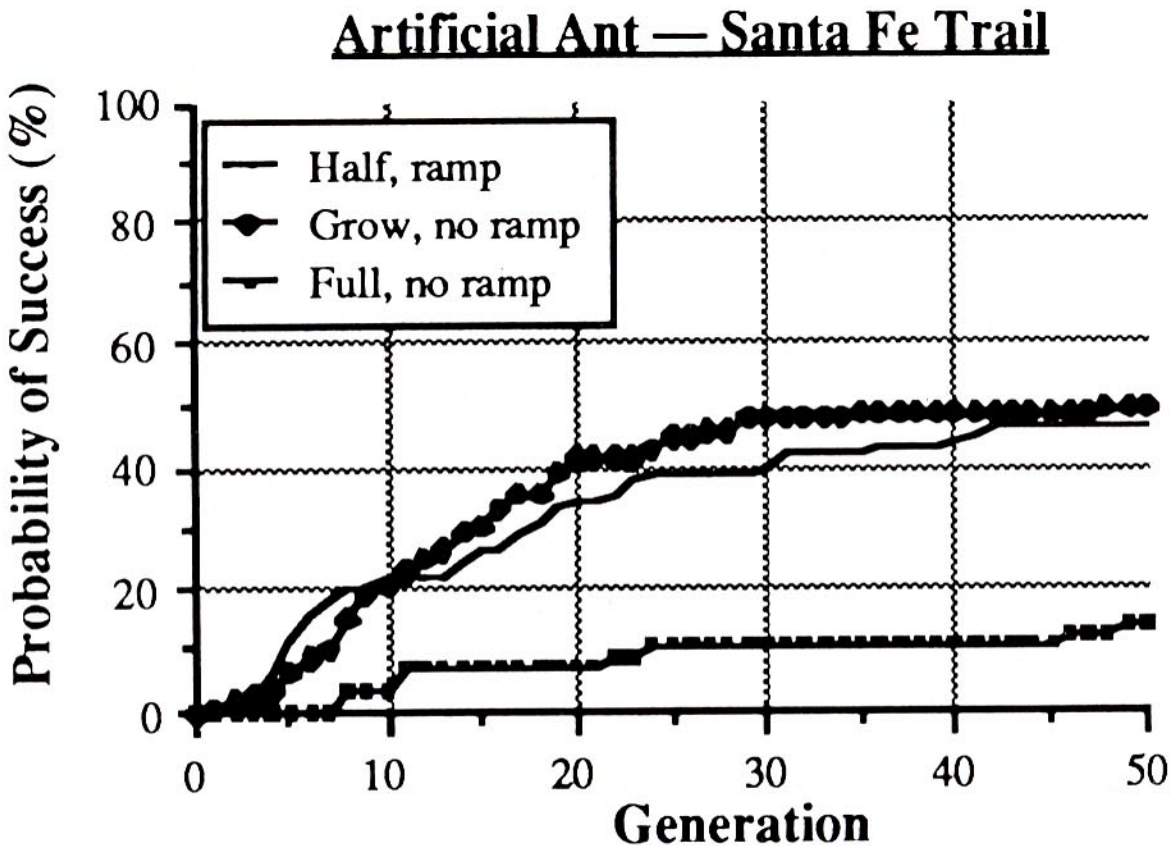


Figure 25.3 Comparison of three methods of creating the initial random population on artificial ant problem.

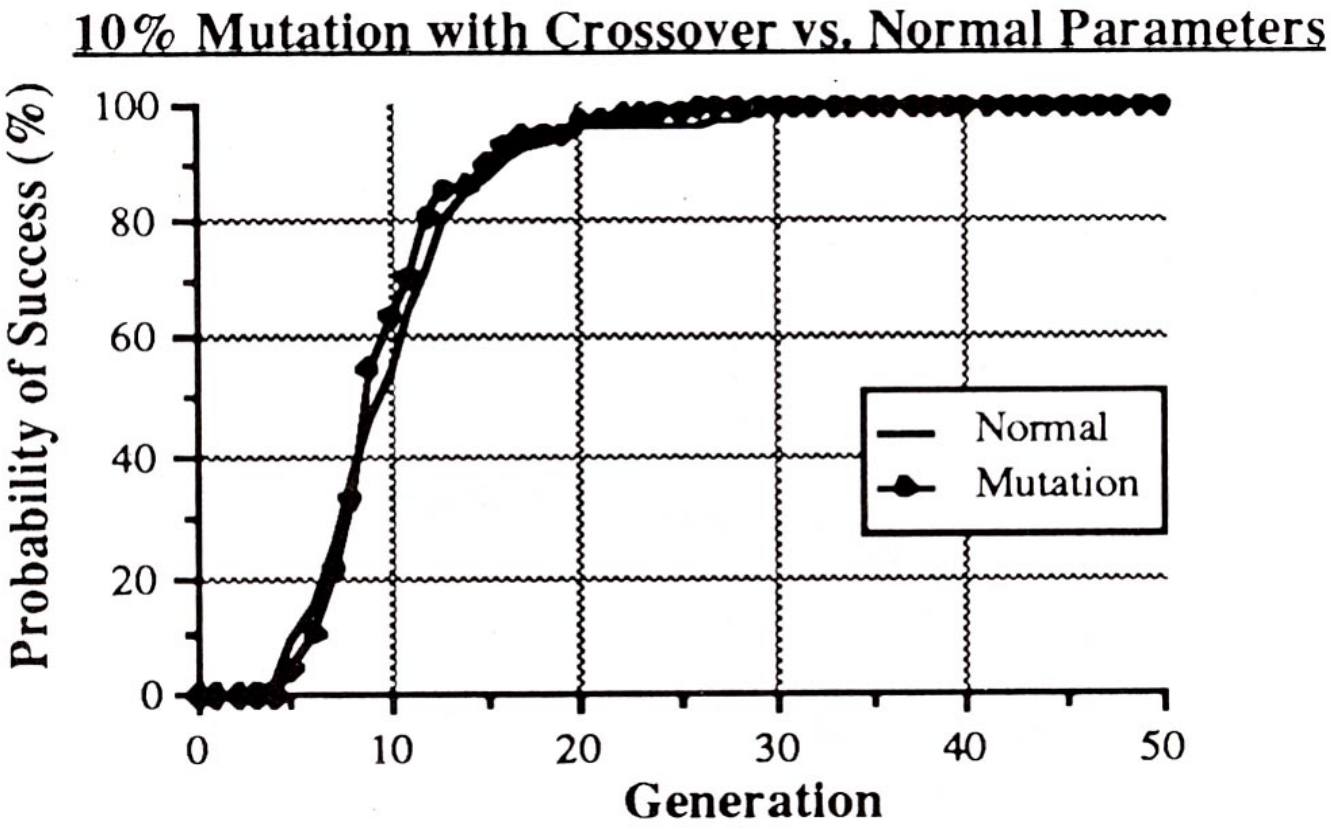


Figure 25.5 Effect of the mutation operation.

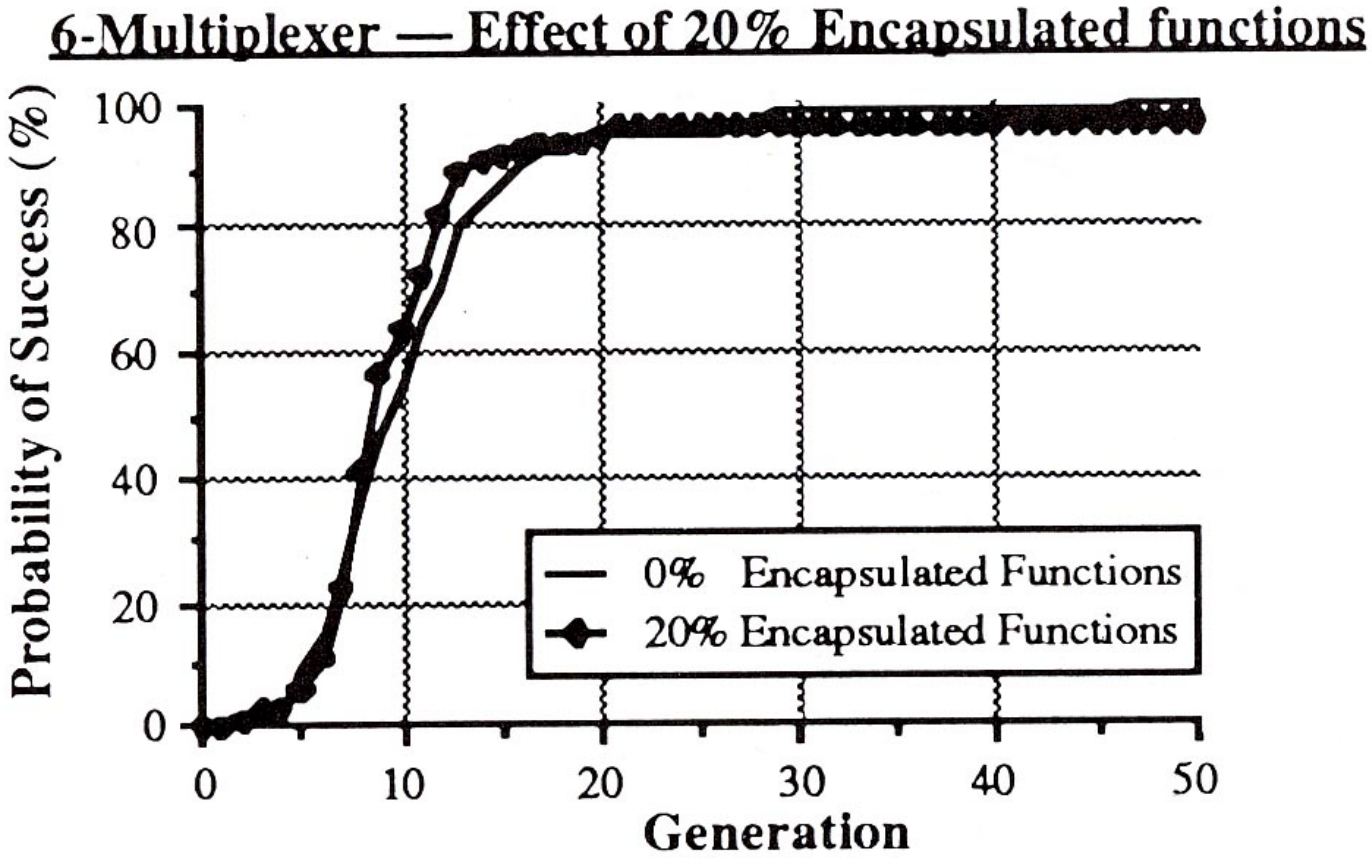


Figure 25.7 Effect of the encapsulation operation.