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.)*

bigger search space

handout 11-18-11

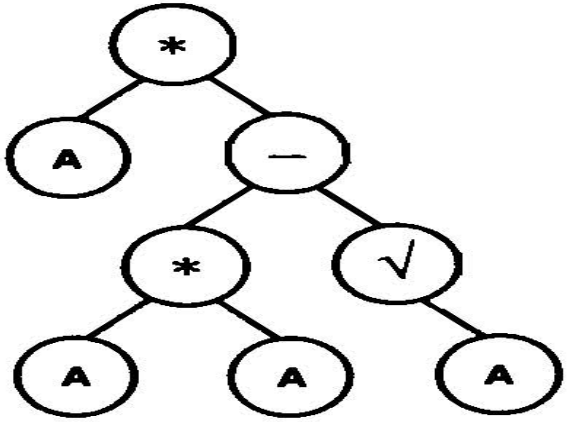
Step 2

- **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.)*

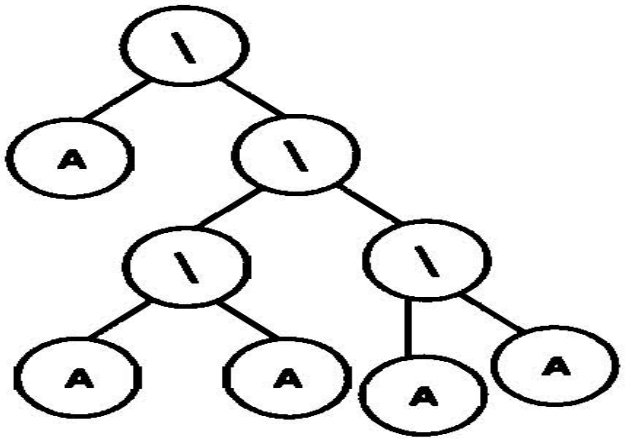
- *Notice randomly generated programs can be of different sizes (i.e. can have different numbers of nodes and levels in the trees.)*

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



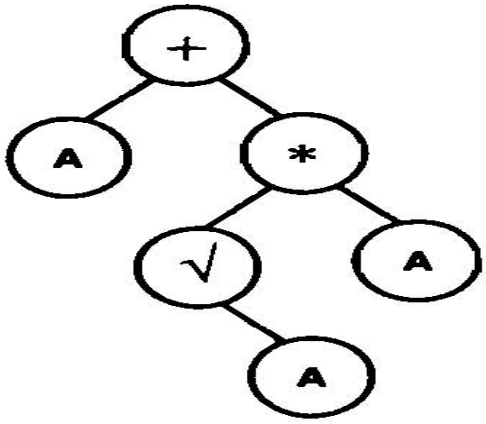
A * [(A * A) – √ A]

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).**

check for correctness or "distance" from being correct

- *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"*

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.*

Steps

1. Choose "functions" and terminals
2. Initial Population
3. Fitness
4. Generate New Population w/ crossover & mutation

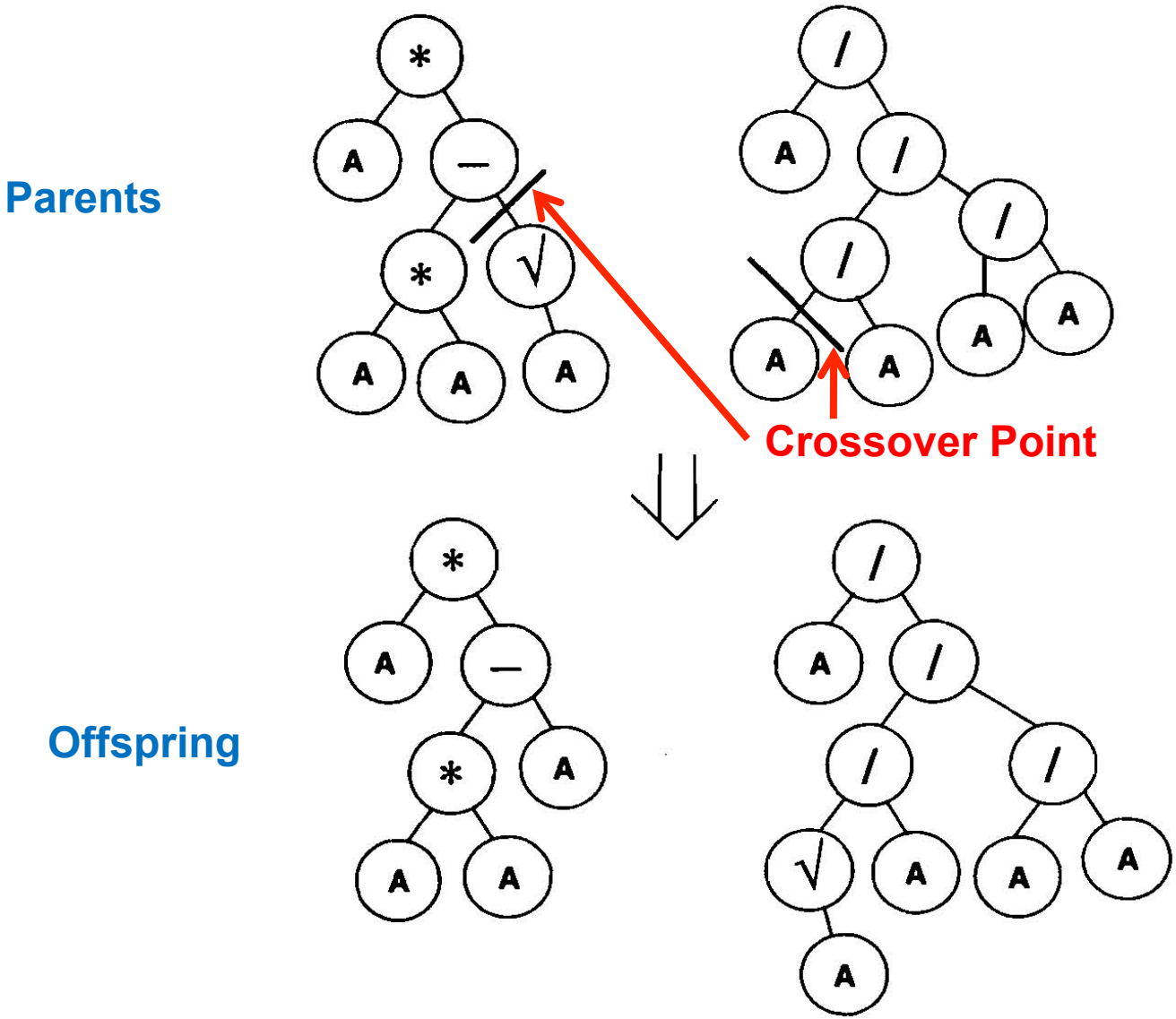
$$y = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

regression finds values for a_3 to a_0.

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.* → How do you mutate a function? + ⇒ (+1)+ ?
  ⇒ changing function is also kinda silly
- ***Step 3 and 4 are repeated until a stopping criterion is satisfied.***

Repeated Slide: Crossover in Genetic Programming



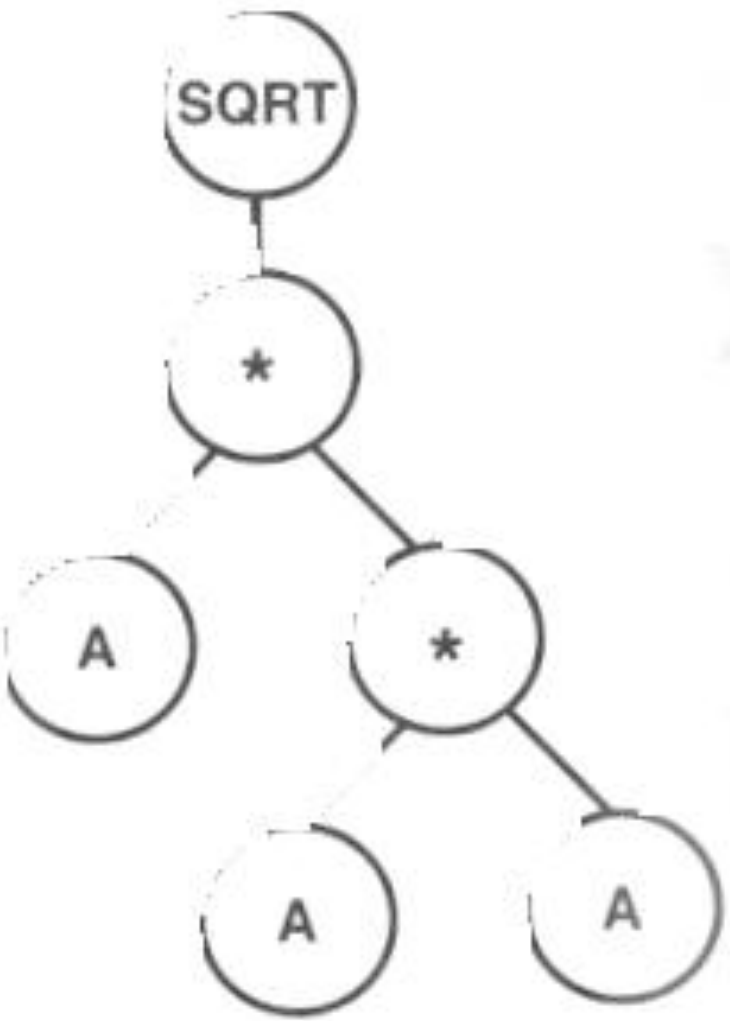


Figure 2.1 Parse tree for the Lisp

- **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 $P = \text{sqrt}(A^3)$ 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
- $P = \text{sqrt}(A^3)$
- The Genetic Program found this as the best solution after some generations.
- *So does that mean we don't need theoretical thinkers?* ⇒ garbage in, garbage out. QED.

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 regression 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, but you assume that you know the form of the equation in regression and that you are looking only for the coefficient values. (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 $f(x)= \sin(x)/(\cos(x) + 12x + x^2$**

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?

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 data to function regression. 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 4^{th} 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

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

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: $+, -, \times, /, \sin, \cos, \exp, r\log$

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, over 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 - Same as raw fitness for this problem

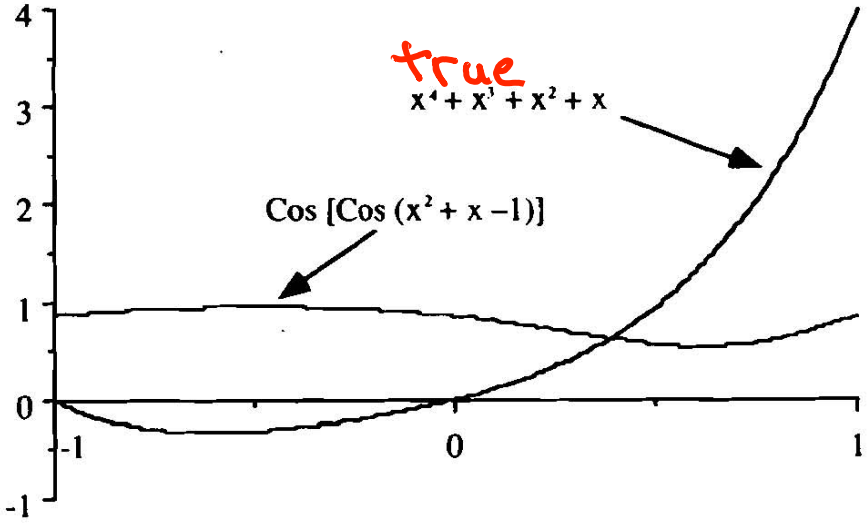
Hits: Number of fitness cases for which the value of the dependent variable produced by the S-expression comes within 0.01 of the target value y_i of the dependent variable

Wrapper: NONE

Parameters: M = 500 (population) a = 5 (generations) | 1

Success predicate: An S-expression scores 20 hits

Δ score with 0.01 std dev 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.

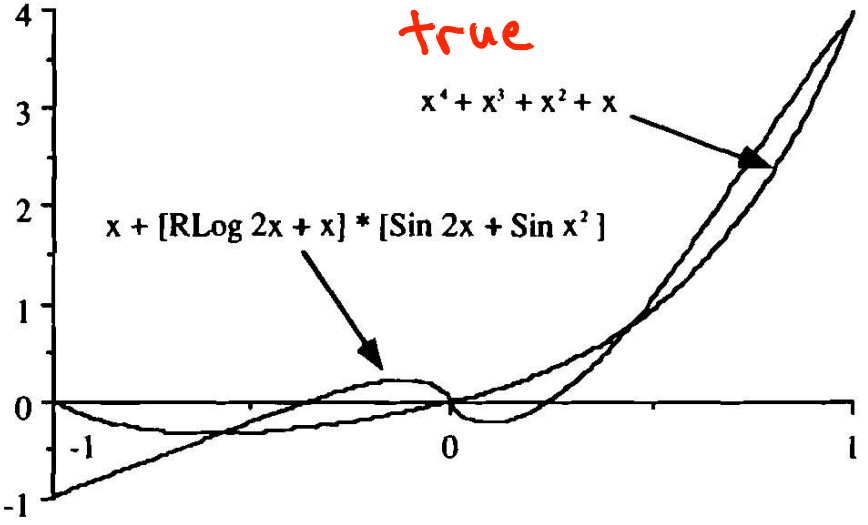


Figure 7.23 Second-best individual from generation 0 compared to target quartic curve $x^4 + x^3 + x^2 + x$ for the simple symbolic regression problem.

Best of Generation from Generation 0

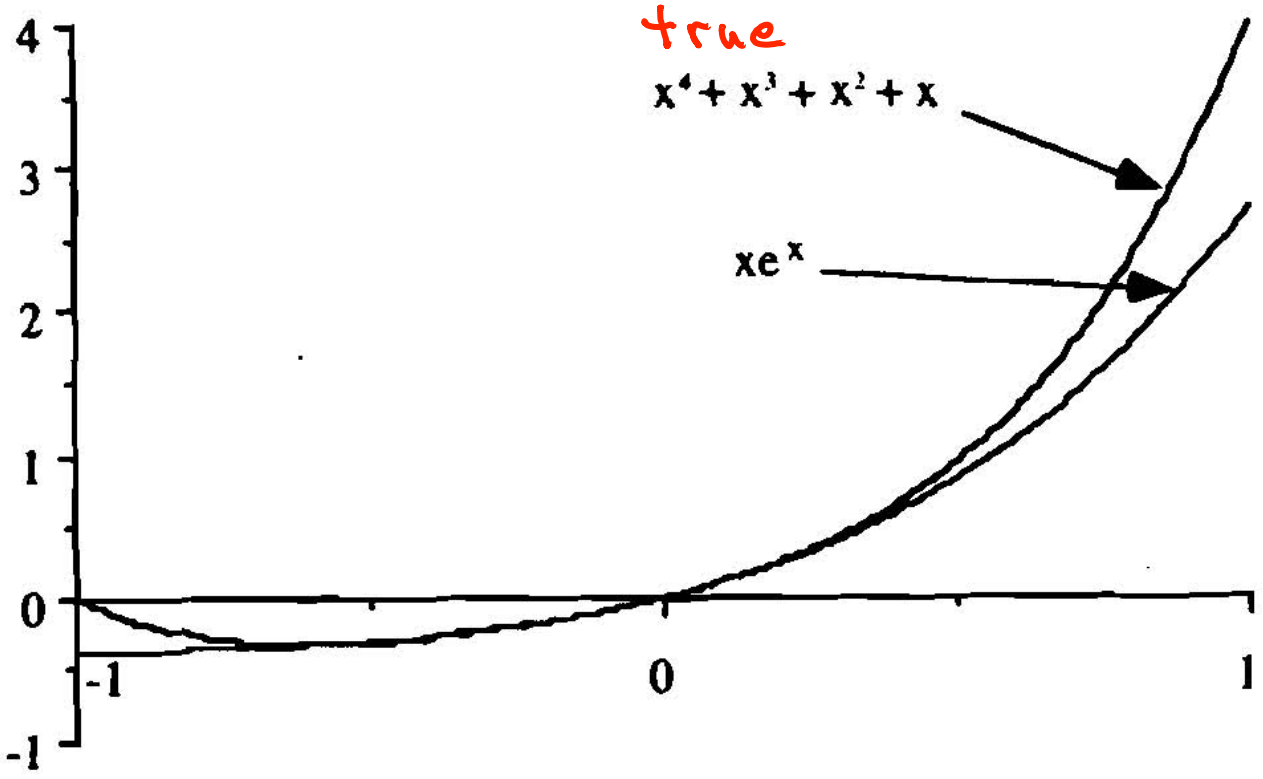


Figure 7.24 Best-of-generation individual from generation 0 compared to target quartic curve $x^4 + x^3 + x^2 + x$ for the simple symbolic regression problem.

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.

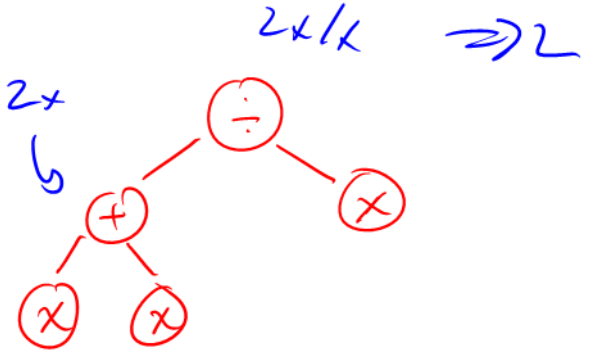
		0	1	2	3	4
1	x_i	−1.0	−0.5	.00	+.5	+1.0
2	$y = xe^x$	−.368	−.303	.000	.824	2.718
3	T	0.0	−.312	.000	.938	4.0
4	$\|T - y\|$	.368	.009	.000	.113	1.212

Y is the best of generation 0.

T= s(x). You compute fitness by adding all the numbers in 4th row (but there would be 20 values not just the 5 values shown here)

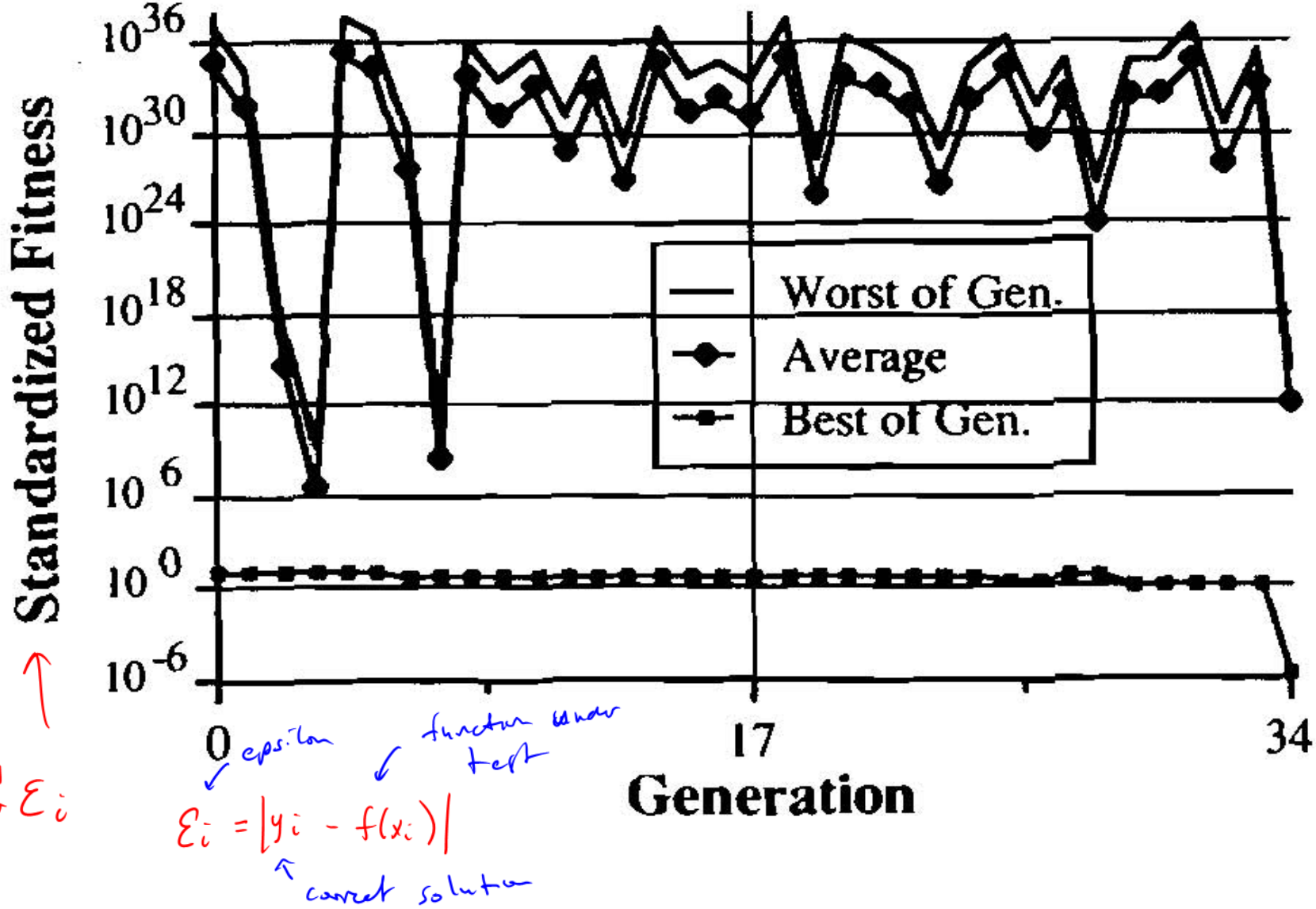
Best of Generation 2

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



- 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?

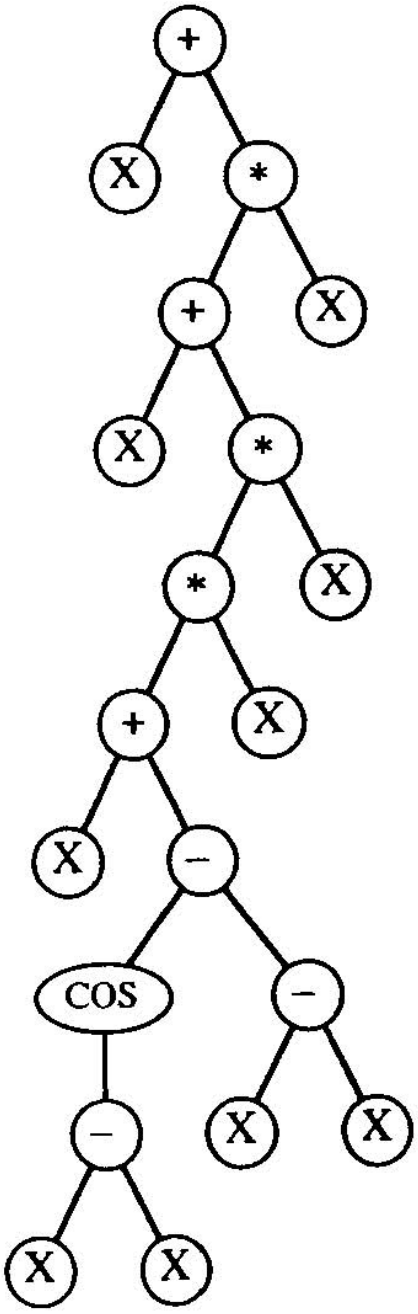
Fitness Curves for the simple symbolic regression problem



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?



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.

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?