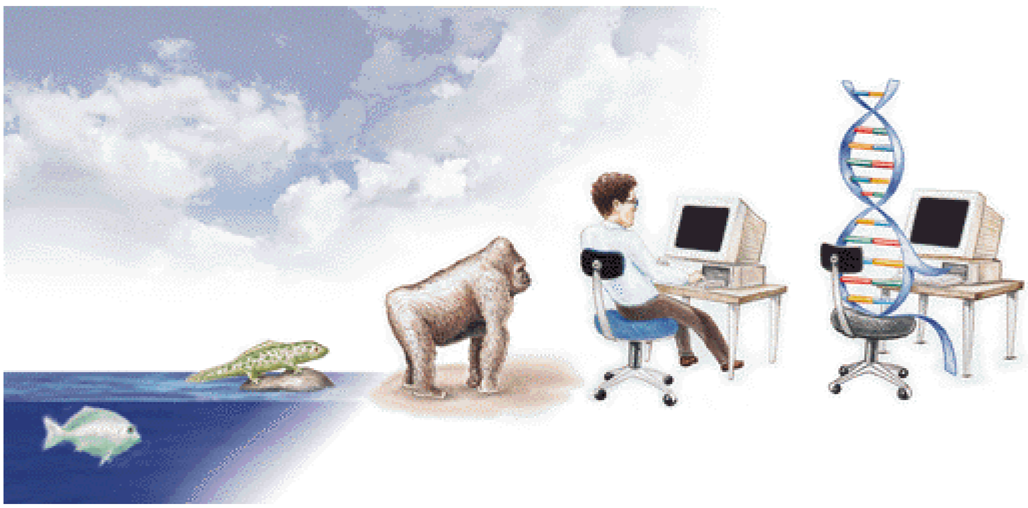
INTRODUCTION TO GENETIC PROGRAMMING



AUTOMATIC PROGRAMMING

- **Program synthesis**
- **Program induction**

THE PROBLEM OF AUTOMATIC PROGRAMMING

"How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what is needed to be done, without being told exactly how to do it?"

---Attributed to Arthur Samuel - about 1959

GENETIC PROGRAMMING (GP)

"Genetic programming *is* automatic programming. For the first time since the idea of automatic programming was first discussed in the late 40's and early 50's, we have a set of non-trivial, non-tailored, computer-generated programs that satisfy Samuel's exhortation: 'Tell the computer what to do, not how to do it.' "

– John Holland, University of Michigan, 1997

A PROGRAM IN THE PASCAL PROGRAMMING LANGUAGE

```
function foo(time:integer)
   :integer;
begin
 temp integer;
 if (time > 10) then temp := 3;
                else temp := 4;
 foo := 1 + 2 + temp;
end;
```

A PROGRAM IN C

```
int foo (int time)
{
   int temp1, temp2;
   if (time > 10)
       temp1 = 3;
   else
       temp1 = 4;
   temp2 = temp1 + 1 + 2;
   return (temp2);
}
```

A PROGRAM IN THE LISP PROGRAMMING LANGUAGE

```
(+ 1 2 (IF (> TIME 10) 3 4))
```

PROGRAM IN LISP = S-EXPRESSION = PARSE TREE = PROGRAM TREE = DATA = LIST

- **TERMINALS = {1, 2, 10, 3, 4, TIME}**
- **FUNCTIONS = {+, IF, >}**
- **S-EXPRESSIONS =**

```
(> TIME 10)
(IF (> TIME 10) 3 4)
(+ 1 2 (IF (> TIME 10) 3 4))
```

GENETIC PROGRAMMING

(1) Generate an initial population of compositions (typically random) of the functions and terminals of the problem.

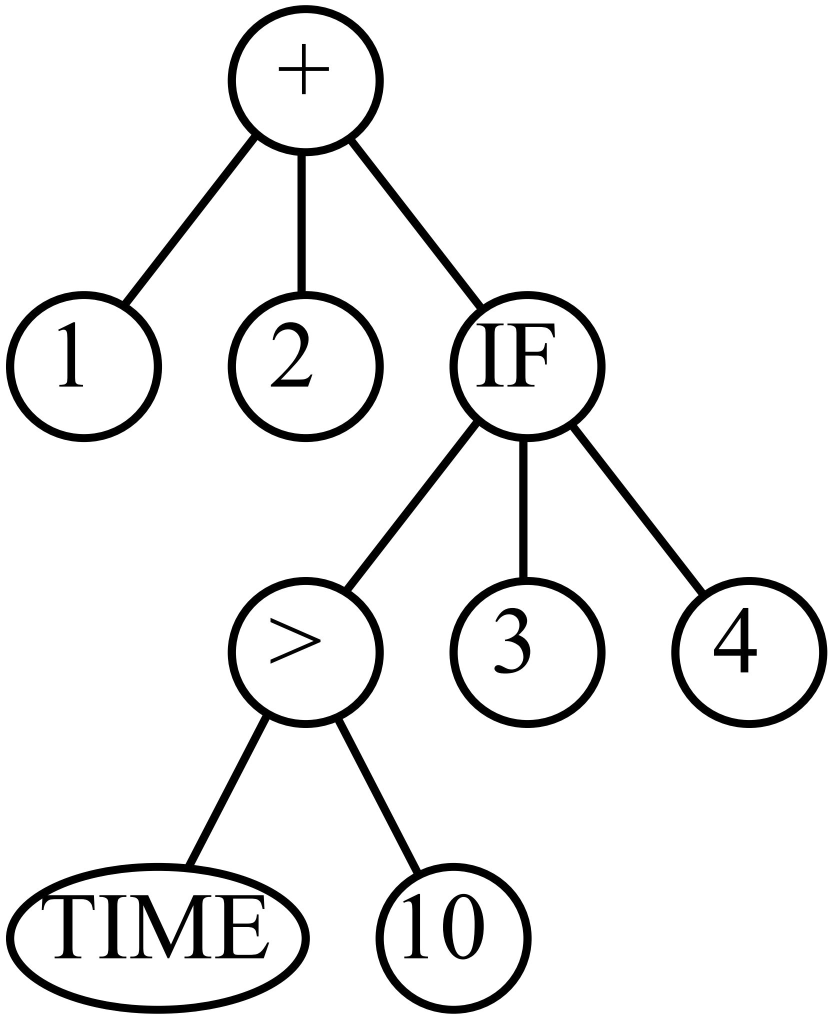
(2) Iteratively perform the following substeps (referred to herein as a generation) on the population of programs until the termination criterion has been satisfied:

(A) Execute each program in the population and assign it a fitness value using the fitness measure.

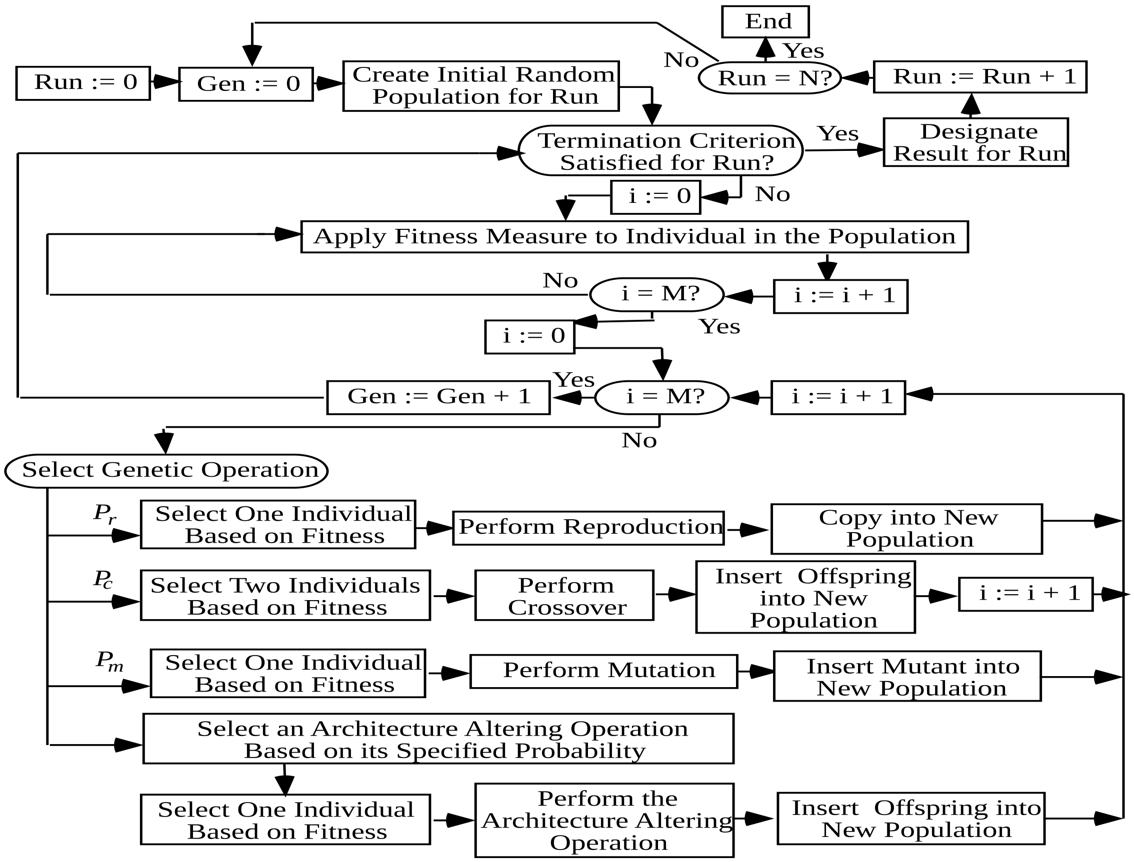
(B) Create a new population of programs by applying the following operations. The operations are applied to program(s) selected from the population with a probability based on fitness (with reselection allowed).

(i) Reproduction
(ii) Crossover (Sexual recombination)
(iii) Mutation
(iv) Architecture-altering operations

(3) Designate the individual program that is identified by result designation (e.g., the best-so-far individual) as the result of the run of genetic programming. This result may be a solution (or an approximate solution) to the problem.


+
1
2
IF
>
3
4
TIME
10

GP FLOWCHART

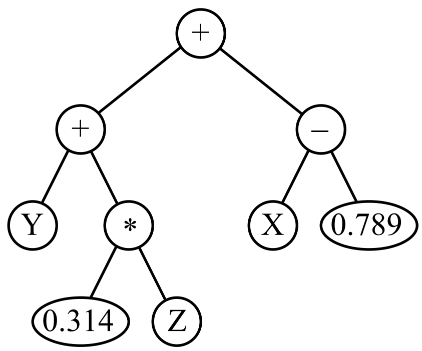


ELEMENTS OF GP FLOWCHART

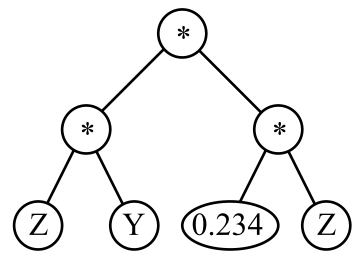
- **Creation of the initial population (generation 0)**
- **Evaluate fitness of each individual in the population for the current generation**
- **Select genetic operation**
  - **reproduction**
  - **mutation**
  - **crossover (recombination) (can be 1-offspring or 2-offspring version)**
  - **architecture-altering operations**
- **Select one or two individuals from the population probabilistically based on fitness**
- **Perform the genetic operation**
- **Insert offspring into population**
- **Termination criterion**
- **Results designation**

INITIAL RANDOM POPULATION OF COMPUTER PROGRAMS

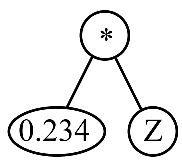
- **Terminal set T = {`X`, `Y`, `Z`, $\Re$}**
- **Function set F = {`+`, `–`, `*`, `%`, `IFLTE`}**



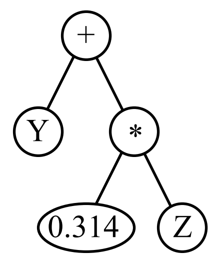
$Y + 0.314Z + X - 0.789$



$0.234Z^2Y$



$0.234Z$

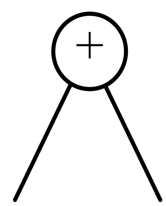


$Y + 0.314Z$

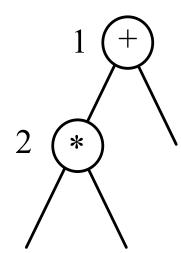
EXAMPLE OF THE CREATION OF A RANDOM PROGRAM TREE

- **Terminal set T = {A, B, C}**
- **Function set F = {+, –, *, %, IFLTE}**

- **Randomly choose a function or terminal from the combined set {+, –, *, %, IFLTE, A, B, C}. Suppose it the two-argument addition (+) function.**

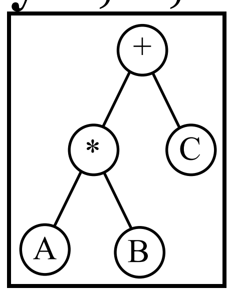


- **Randomly choose another function or terminal from {+, –, *, %, IFLTE, A, B, C}, say the two-argument multiplication (*) function.**



EXAMPLE OF THE CREATION OF A RANDOM PROGRAM TREE – CONTINUED

- **Continue in this manner. Suppose that the next 3 random choices from {+, −, *, %, IFLTE, A, B, C}, say A, B, and C.**



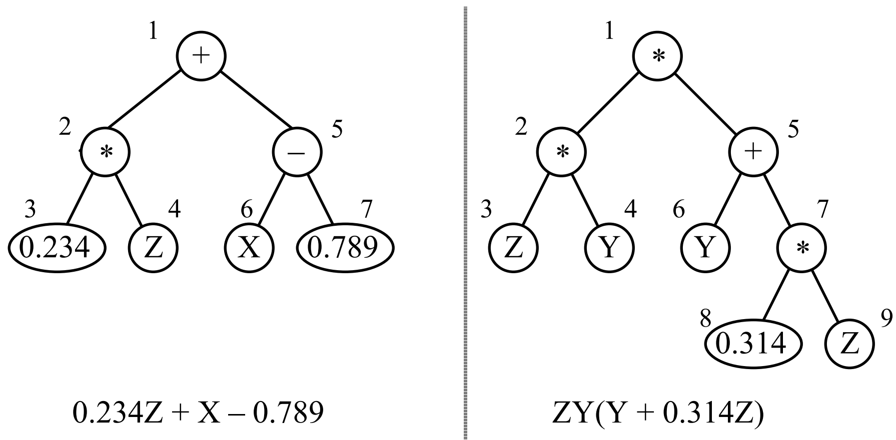
- **The growth process ends when all paths end in a terminal {A, B, C}.**
- **Force a choice from the terminal set (rather than the combined set) if the preestablished maximum size (measured in terms of number of functions and terminals or in terms of depth) is being exceeded.**

CROSSOVER (SEXUAL RECOMBINATION) OPERATION FOR COMPUTER PROGRAMS (PROGRAM TREES)

- **Select two parents chosen based on fitness**

- **Randomly pick a number from 1 to `NUMBER-OF-POINTS` – independently for each parental program**

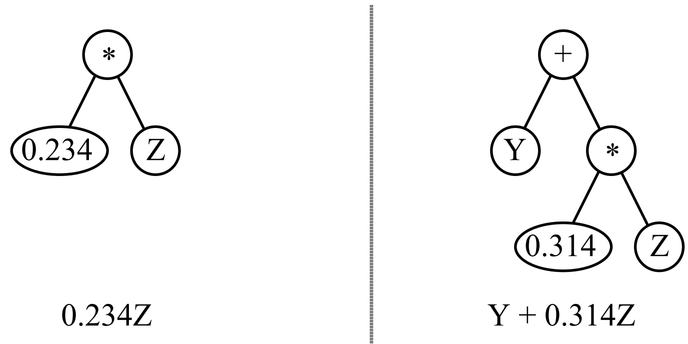
TWO PARENTS IN CROSSOVER



(+ <u>(* 0.234 Z)</u> (- X 0.789))

(* (* Z Y) <u>(+ Y (* 0.314 Z))</u>)

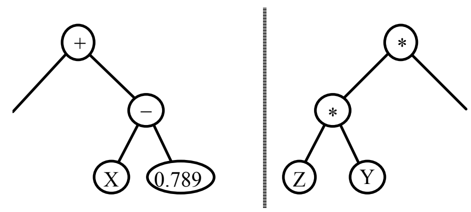
CROSSOVER FRAGMENTS



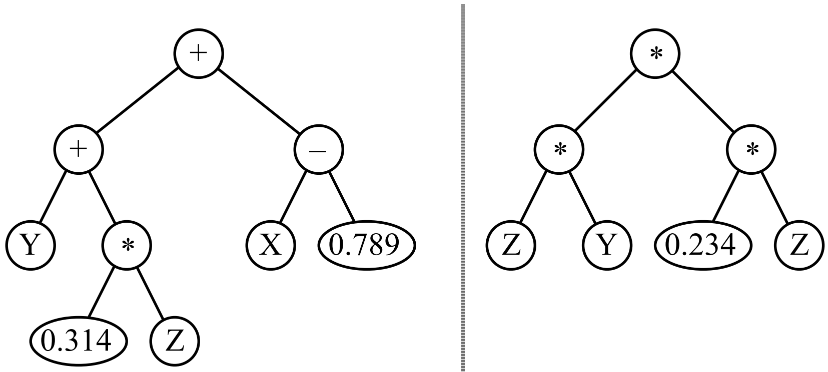
(+ <u>(* 0.234 Z)</u> (- X 0.789))

(* (* Z Y) <u>(+ Y (* 0.314 Z))</u>)

TWO REMAINDERS



TWO OFFSPRING



$Y + 0.314Z + X - 0.789$ $0.234Z^2Y$

```
(+ (+ Y (* 0.314 Z))
   (- X 0.789))
```

```
(* (* Z Y) (* 0.234 Z))
```

THE CROSSOVER OPERATION PRODUCES SYNTACTICALLY VALID, EXECUTABLE COMPUTER PROGRAMS

IF SET OF FUNCTIONS AND TERMINALS IS CLOSED (I.E., ANY FUNCTION CAN ACCEPT THE OUTPUT PRODUCED BY ANY OTHER FUNCTION OF TERMINAL)

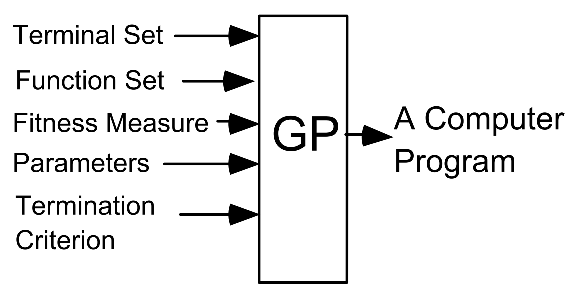
- **Protected division % takes two arguments and returns one when division by 0 is attempted (including 0 divided by 0), and, otherwise, returns the normal quotient**
- **Protected multiplication, addition, and subtraction**

MUTATION OPERATION FOR COMPUTER PROGRAMS (PROGRAM TREES)

- **Select parent based on fitness**
- **Pick point**
- **Delete subtree at that point**
- **Grow new subtree at the mutation point in the same way as generated trees for initial random population (generation 0)**

GP – 5 MAJOR PREPARATORY STEPS

- **determining the set of terminals**
- **determining the set of functions**
- **determining the fitness measure**
- **determining the parameters**
  - **population size**
  - **number of generations**
- **determining the method for designating a result and the criterion for terminating a run**



SYMBOLIC REGRESSION

Independent variable *X*	Dependent Variable *Y*
-1.0	1.00
-0.9	0.91
-0.8	0.84
-0.7	0.79
-0.6	0.76
-0.5	0.75
-0.4	0.76
-0.3	0.79
-0.2	0.84
-0.1	0.91
0.0	1.00
0.1	1.11
0.2	1.24
0.3	1.39
0.4	1.56
0.5	1.75
0.6	1.96
0.7	2.19
0.8	2.44
0.9	2.71
1.0	3.00

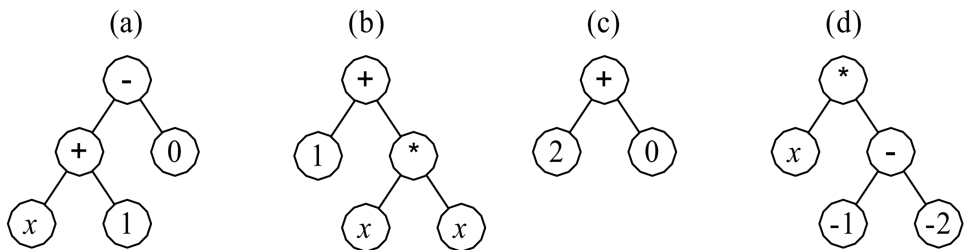
TABLEAU FOR SYMBOLIC REGRESSION

	Objective:	**Find a computer program with one input (independent variable *x*), whose output equals the observed data in range from -1 to +1.**
1	**Terminal set:**	`T = {X, Random-Constants}`
2	**Function set:**	`F = {+, -, *, %}` **NOTE: The protected division function `%` returns a value of 1 when division by 0 is attempted (including 0 divided by 0)**
3	**Fitness:**	**The sum of the absolute value of the differences (errors), over the values of the independent variable *x* from –1.0 to +1.0, between the program's output and the observed data.**
4	**Parameters:**	**Population size *M* = 4.**

5	**Termination:**	**An individual emerges whose sum of absolute errors is less than 0.1**

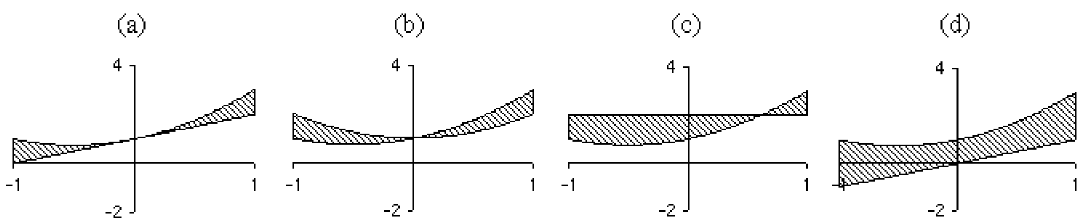
SYMBOLIC REGRESSION OF QUADRATIC POLYNOMIAL $X^2 + X + 1$

INITIAL POPULATION OF FOUR RANDOMLY CREATED INDIVIDUALS OF GENERATION 0



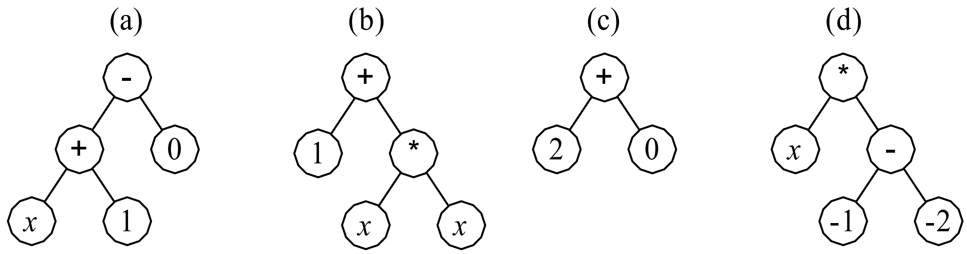
$X + 1$ $X^2 + 1$ 2 X

FITNESS

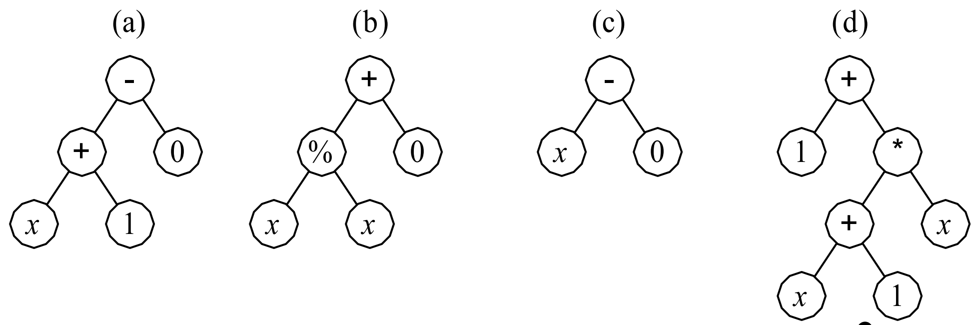


0.67 **1.00** **1.70** **2.67**

SYMBOLIC REGRESSION OF QUADRATIC POLYNOMIAL $X^2 + X + 1$

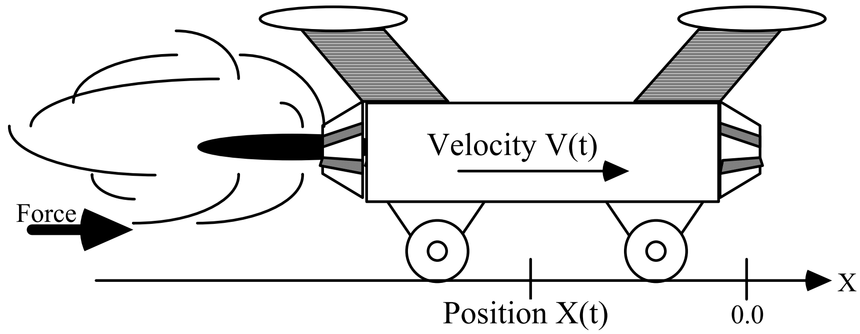


GENERATION 1



$x + 1$	1	X	$x^2 + x + 1$
Copy of (a)	**Mutant of (c)**	**First offspring of crossover of (a) and (b)**	**Second offspring of crossover of (a) and (b)**

CART CENTERING PROBLEM (ISOTROPIC ROCKET)



- **A bang-bang force (+1, =1) is applied to the cart at position x(t) on the frictionless track. The cart has velocity v(t). The applied force produces an acceleration of**

$$a(t) = \frac{F(t)}{m}$$

- **The cart has initial position x(0) and initial velocity v(0)**
- **The new velocity v(t + 1) at time t = t+1 is**

$$v(t + 1) = v(t) + \tau\, a(t)$$

- **The new cart position x(t + 1) is**

$$x(t + 1) = x(t) + \tau\, v(t)$$

CART CENTERING — CONTINUED

• **The goal is to bring the cart to rest at the origin (i.e., near-zero velocity and near-zero position) in minimal time by applying forces of either +1 or -1 at each time step.**

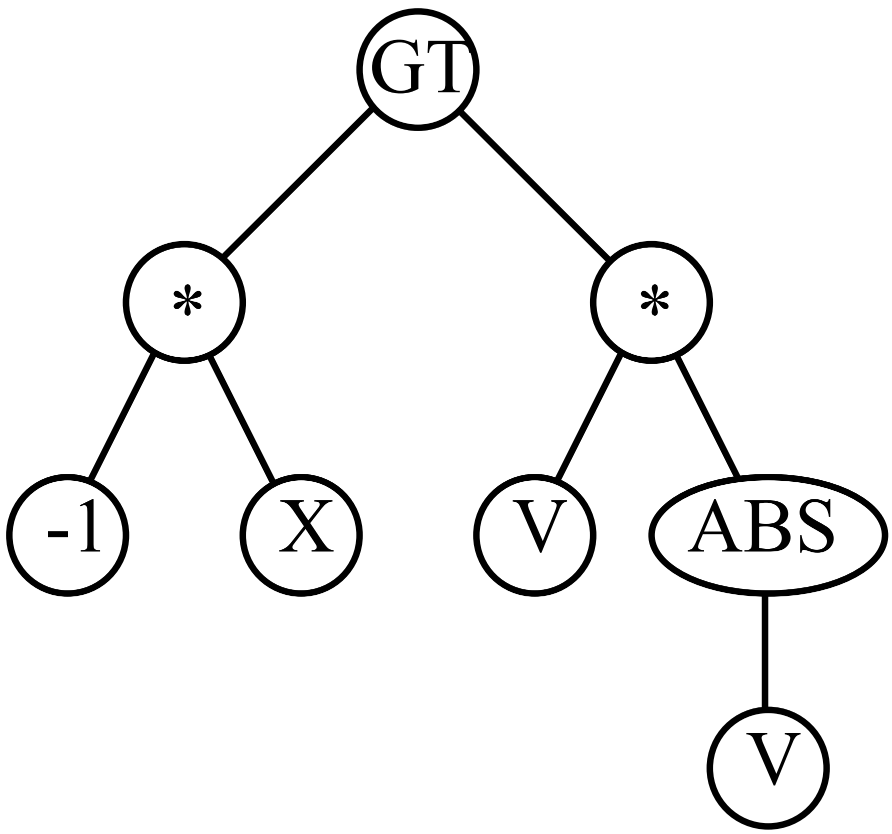
The time-optimal solution is to accelerate cart in the positive direction if

$$- x(t) > \frac{v(t)^2 \text{ Sign } v(t)}{\dfrac{2|F|}{m}}$$

CART CENTERING — CONTINUED

PROGRAM IN LISP

```
(GT (* -1 X) (* V (ABS V)))
```



CART CENTERING — CONTINUED

NON-OPTIMAL STRATEGY NO. 1

$$- x(t) > \frac{v(t)^3}{\frac{2|F|}{m}}$$

```
(GT (* -1 X) (* V (* V V)))
```

CART CENTERING — CONTINUED

NON-OPTIMAL STRATEGY NO. 2

$$-x(t) > \frac{v(t)^2}{\frac{2|F|}{m}}$$

```
(GT (* -1 X) (* V V))
```

CART CENTERING — CONTINUED

FLOWCHART FOR COMPUTING FITNESS FOR ONE INDIVIDUAL OVER N_{FC} FITNESS CASES, EACH INVOLVING A SIMULATION OVER T_{MAX} TIME STEPS

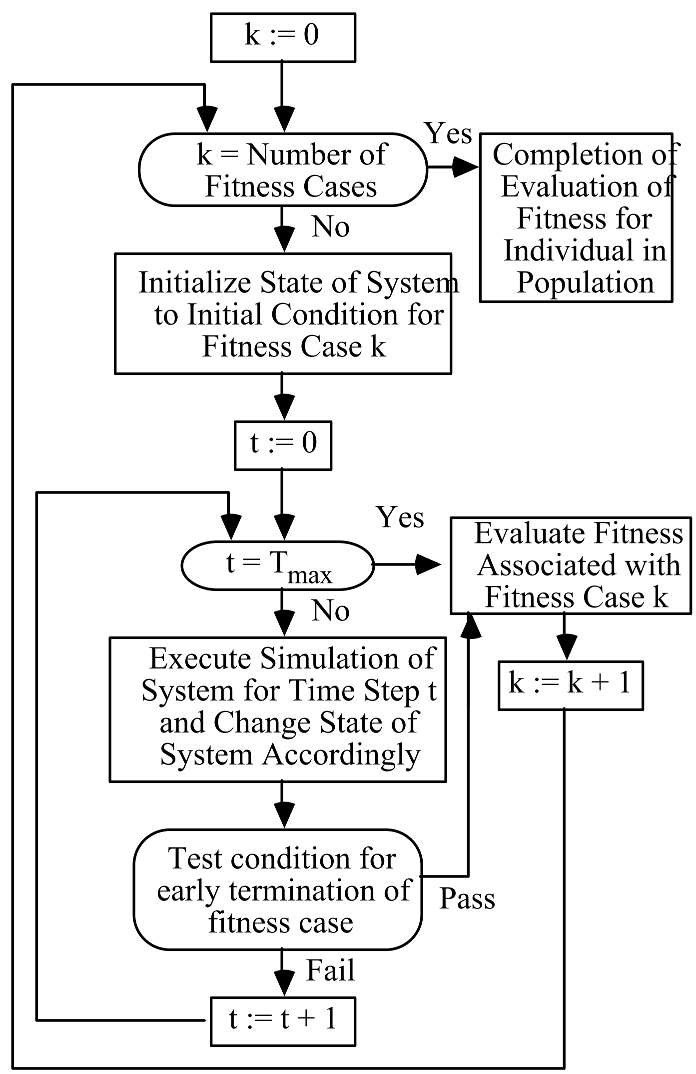


TABLEAU FOR THE CART CENTERING

Objective:	**Find a time-optimal bang-bang control strategy to center a cart on a one-dimensional track.**
Terminal set:	**The state variables of the system: *x* (position X of the cart) and *v* (velocity V of the cart).**
Function set:	`+, -, *, %, ABS, GT.`
Fitness cases:	**20 initial condition points (*x*, *v*) for position and velocity chosen randomly from the square in position-velocity space whose opposite corners are (–0.75, 0.75) and (0.75,–0.75).**
Raw fitness:	**Sum of the time, over the 20 fitness cases, taken to center the cart. When a fitness case times out, the contribution is 10.0 seconds.**
Standardized fitness:	**Same as raw fitness for this problem.**

Hits:	**Number of fitness cases that did not time out.**
Wrapper:	**Converts any positive value returned by an S-expression to +1 and converts all other values (negative or zero) to –1.**
Paramete rs:	M **= 500.** G **= 51.**
Success Predicate:	**None.**

CART CENTERING — CONTINUED

GENERATION 0

- **Many apply acceleration relentlessly in positive (or negative) direction**

```
(* (* V X) (* V X))
```

- **Many are partially blind**

```
(+ V V)
```

- **14% of the 500 individuals in the population time-out for all 20 fitness cases.**
- **44% time-out for all but one fitness case**

CART CENTERING — CONTINUED

GENERATION 0

• Average fitness is 187.4 seconds (9.37 seconds per fitness case)

• Third-best control strategy equivalent to

```
(- X (+ V (* 2 V X)))
```

• Centers the cart in less than 10 seconds for all 20 fitness cases. Takes 178.6 seconds (an average of 8.93 seconds per fitness case). Equivalent to

Sign (x - v –2vx)

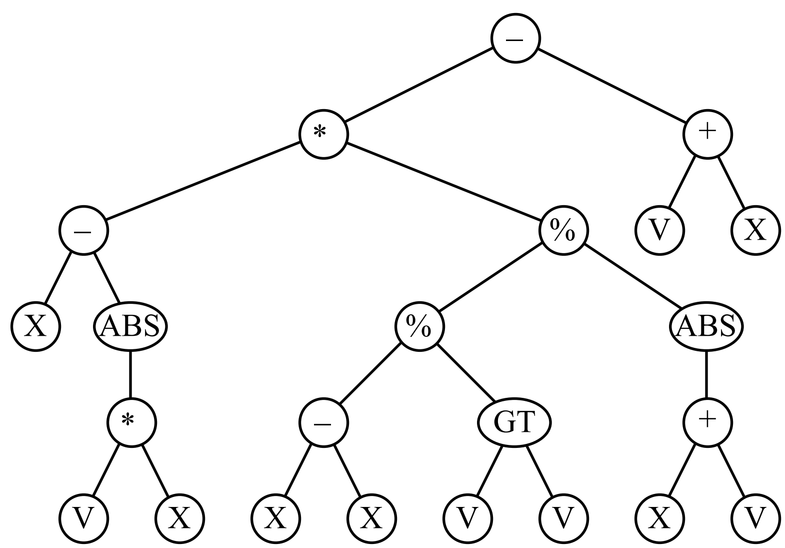
• The second-best control strategy takes 130.0 seconds (an average of 6.05 seconds per fitness case).

CART CENTERING — CONTINUED

BEST-OF-GENERATION 0

- **Takes only 48.6 seconds (an average of 2.43 seconds per fitness case).**

```
(- (* (- X (ABS (* V X)))
      (% (% (- X X) (GT V V))
         (ABS (+ X V))))
   (+ V X))
```



- **Equivalent to straight line with slope –45°:**

```
(- 0 (+ V X))
```

CART CENTERING — CONTINUED

BEST-OF-GENERATION 3

- **Takes an average of 2.24 seconds per fitness case.**

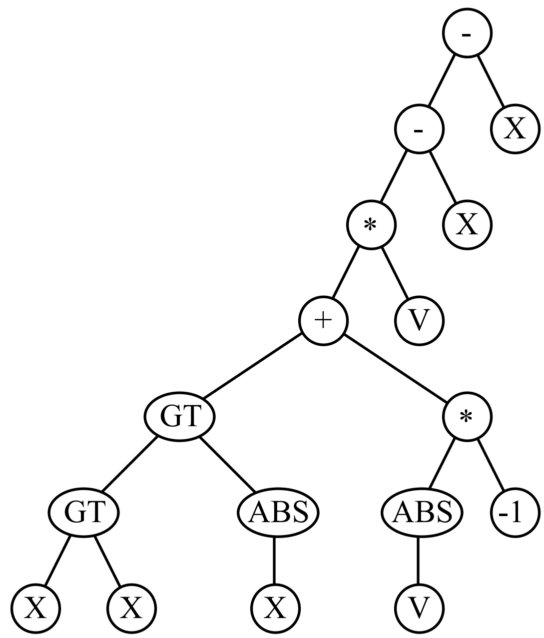
```
(- (- (* (+ (GT (GT X X) (ABS X))
                (* (ABS V) -1))
             V)
          X)
      X)
```

- **Equivalent, for the range of x being used here, to**

$$-v[1+|v|]-2x$$

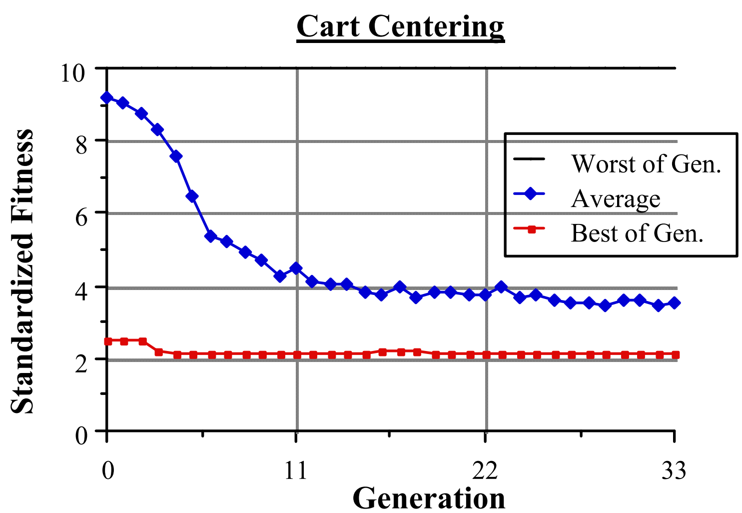
CART CENTERING — CONTINUED

BEST-OF-GENERATION 3



CART CENTERING — CONTINUED

FITNESS CURVES



CART CENTERING — CONTINUED

BEST-OF-RUN INDIVIDUAL FROM GENERATION 33

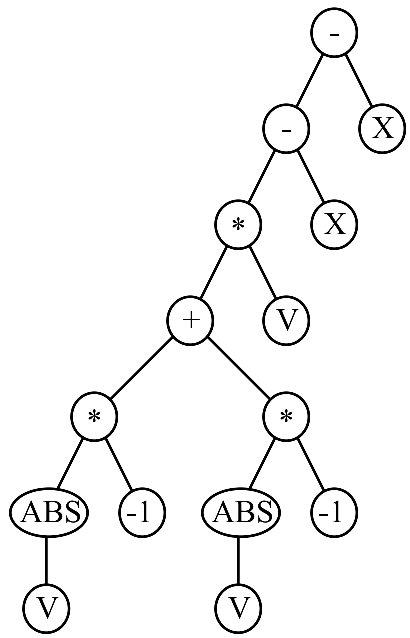
```
(- (- (* (+ (* (ABS V) -1) (*
(ABS V) -1))
          V)
      X)
   X)
```

• **100%-correct solution – Equivalent to the known time-optimal solution**

```
(GT (* -1 X) (* V (ABS V)))
```

CART CENTERING — CONTINUED

BEST-OF-RUN INDIVIDUAL FROM GENERATION 33



CART CENTERING — CONTINUED

DIFFERENT, BUT EQUIVALENT SOLUTIONS – CART-CENTERING PROBLEM

```
(GT (% V (% -1 (ABS V))) X)
```

```
(GT (* (GT (* -1 X) X) (ABS X))
    (* (ABS V) V))
```

```
(GT -1 (% (+ (GT (- V -1) (- -1
V)) (ABS (GT (% (+ (GT (- V -1)
(- -1 V)) (ABS (+ (+ V (+ X V))
(% X X)))) (GT V (% (% (* X -1)
(% (- -1 V) (GT V (* X -1))))
(* -1 -1)))) -1))) (GT V (% (*
X -1) (ABS V)))))
```

CART CENTERING — CONTINUED

NEAR-OPTIMAL / APPROXIMATE SOLUTIONS – CART-CENTERING PROBLEM

- **Requires 100.45% of the optimal time:**

```
(+ (GT (* (+ (GT (* (ABS (GT V
(GT V V))) (* X -1)) (GT (+ V
(* V V)) (ABS V))) X) (* V V))
X) (- (+ (GT (GT (* (+ (* (* X
X) (+ -1 X)) (GT V V)) (+ V X))
X) X) (* (* X V) V)) V))
```

- **Requires 100.5% of the optimal time:**

```
(GT (* (+ (+ (GT -1 V) (- -1
X)) (* (+ (+ (+ (% (- X (GT V
(- -1 X))) (* -1 (GT V (ABS
X)))) (GT X (* (ABS X) V))) (-
-1 X)) (* X (* (% (+ -1 X) (GT
(GT (GT V (% X -1)) X) (* (% X
-1) V))) V))) V)) X) V)
```