# *GENETIC PROGRAMMING: THE MOVIE*

# 24 ITEMS IN *GENETIC PROGRAMMING: THE MOVIE*

- **Symbolic Regression**
- **Intertwined Spirals**
- **Artificial Ant**
- **Truck Backer Upper**
- **Broom Balancing**
- **Wall Following**
- **Box Moving**
- **Discrete Pursuer-Evader**
- **Differential Pursuer-Evader**
- **Co-Evolution**
- **Inverse Kinematics**
- **Emergent Collecting**
- **Central Place Foraging**
- **Block Stacking**
- **Randomizer**
- **1-D Cellular Automata**
- **2-D Cellular Automata**
- **Task Prioritization**
- **Programmatic Image Compression**
- **Finding $3\sqrt{2}$**
- **Econometric Exchange Equation**
- **Optimization (Lizard)**
- **Boolean 11-Multiplexer**
- **11-Parity–Automatically Defined Functions**

# REGRESSION PROBLEM

| Independent variable *X* | Dependent Variable *Y* |
|---|---|
| -1.0 | 0.0 |
| -0.9 | -0.1629 |
| -0.8 | -0.2624 |
| -0.7 | -0.3129 |
| -0.6 | -0.3264 |
| -0.5 | -0.3125 |
| -0.4 | -0.2784 |
| -0.3 | -0.2289 |
| -0.2 | -0.1664 |
| -0.1 | -0.0909 |
| 0 | 0.0 |
| 0.1 | 0.1111 |
| 0.2 | 0.2496 |
| 0.3 | 0.4251 |
| 0.4 | 0.6496 |
| 0.5 | 0.9375 |
| 0.6 | 1.3056 |
| 0.7 | 1.7731 |
| 0.8 | 2.3616 |
| 0.9 | 3.0951 |
| 1.0 | 4.0000 |

# SYMBOLIC REGRESSION PROBLEM

**Also called**

- **System Identification problem**
- **the "Black Box" problem**
- **Model building**
- **Empirical discovery**
- **Non-parametric regression**
- **Forecasting / Time-series prediction**

- **We seek**
  - **Functional form of a good fit**
  - **Numerical parameters**
  - **Size and shape of the mathematical expression**
- **The fitness measure is error**
  - **Sum, over fitness cases, of absolute error**
  - **Sum, over fitness cases, of squared error**

# TABLEAU FOR SYMBOLIC REGRESSION OF $X^4 + X^3 + X^2 + X$

| | |
|---|---|
| **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:** | **$M = 500$. $G = 51$.** |
| **Success Predicate:** | **An S-expression scores 20 hits.** |

# GENERATION 0 – SIMPLE SYMBOLIC REGRESSION OF $X^4 + X^3 + X^2 + X$

# WORST-OF-GENERATION INDIVIDUAL IN GENERATION 0 WITH RAW FITNESS OF $10^{38}$

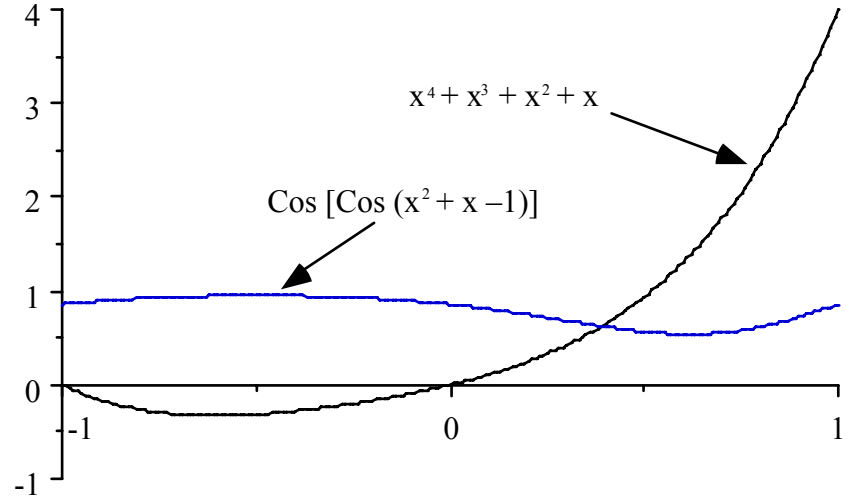
```
(EXP (- (% X (- X (SIN X)))
(RLOG (RLOG (* X X)))))
```

# GENERATION 0 – SIMPLE SYMBOLIC REGRESSION OF $X^4 + X^3 + X^2 + X$ MEDIAN INDIVIDUAL IN GENERATION 0 WITH RAW FITNESS OF 23.67

```
(COS (COS (+ (- (* X X) (% X X)) X)))
```

**equivalent to...**

**Cos [Cos ($x^2$ + x – 1)]**

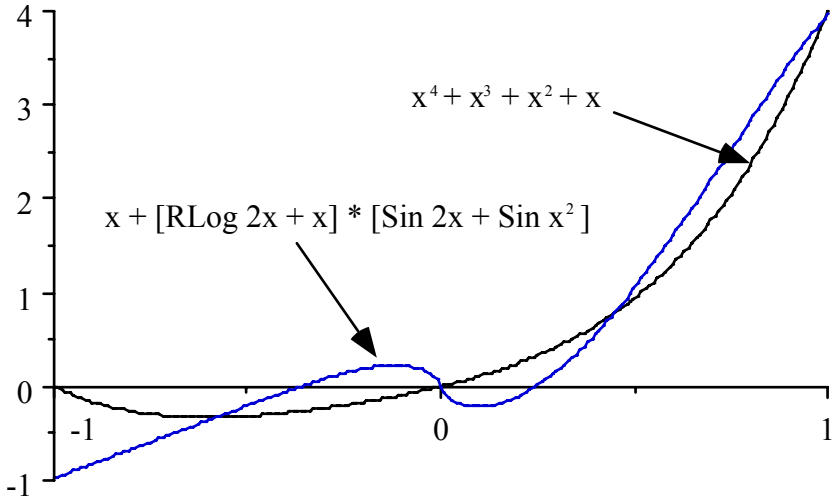



**NOTE: Constant 1 = *x/x***

# GENERATION 0 – SIMPLE SYMBOLIC REGRESSION OF $X^4 + X^3 + X^2 + X$

# SECOND-BEST INDIVIDUAL IN GENERATION 0 WITH RAW FITNESS OF 6.05

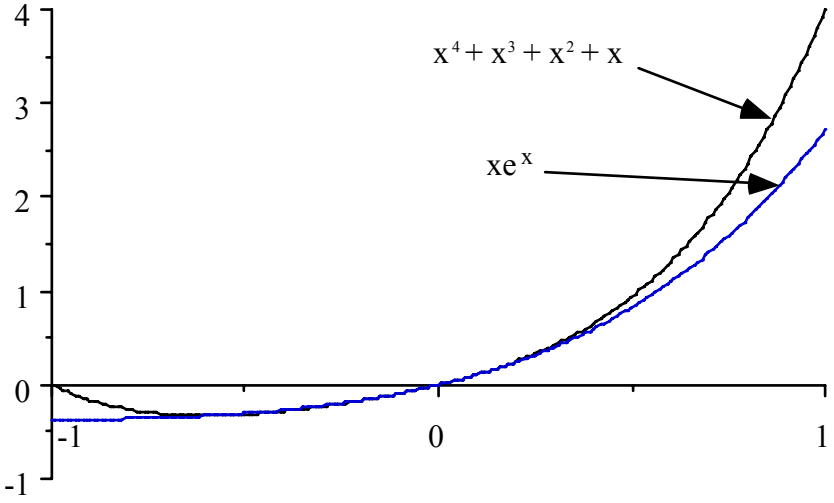
## x + [RLog 2x + x] * [Sin 2x + Sin $x^2$]



# GENERATION 0 – SIMPLE SYMBOLIC REGRESSION OF $X^4 + X^3 + X^2 + X$ BEST-OF-GENERATION INDIVIDUAL IN GENERATION 0 WITH RAW FITNESS OF 4.47

```
(* X (+ (+ (- (% X X) (% X X))
(SIN (- X X)))
          (RLOG (EXP (EXP X)))))
```

**Equivalent to ... $xe^x$**



# CREATION OF GENERATION 1 FROM GENERATION 0

- **In the so-called "generational" model for genetic algorithms, a new population is created that is equal in size to the old population**
  - **1% mutation (i.e., 5 individuals out of 500)**
  - **9% reproduction (i.e., 45 individuals)**
  - **90% crossover (i.e., 225 pairs of parents — yielding 450 offspring)**

# CREATION OF GENERATION 1 FROM GENERATION 0 — CONTINUED

- **All participants in mutation, reproduction, and crossover are chosen from the current population <u>PROBABILISTICALLY, BASED ON FITNESS</u>**
  - **Anything can happen**
  - **Nothing is guaranteed**
  - **The search is heavily (but not completely) biased toward high-fitness individuals**
  - **The best is not guaranteed to be chosen**
  - **The worst is not necessarily excluded**
  - **Some (but not much) attention is given even to low-fitness individuals**

# GENERATION 2 – SIMPLE SYMBOLIC REGRESSION OF $X^4 + X^3 + X^2 + X$

# BEST-OF-GENERATION INDIVIDUAL IN GENERATION 2 WITH RAW FITNESS OF 2.57

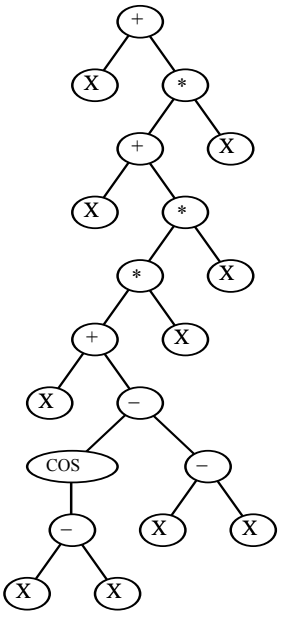
```
(+ (* (* (+ X (* X (* X (% (% X
X) (+ X X)))))
              (+ X (* X X))) X) X)
```

**Equivalent to...**

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

# BEST-OF-RUN INDIVIDUAL IN GENERATION 34 WITH RAW FITNESS OF 0.00 (100%-CORRECT BEST-OF-RUN INDIVIDUAL)

```
(+ X (* (+ X (* (* (+ X (- (COS
(- X X)) (- X X))) X) X)) X))
```



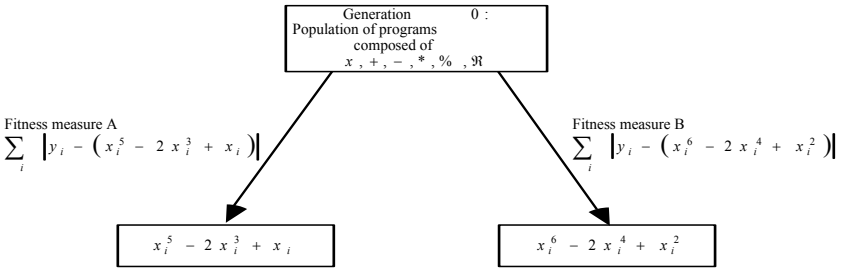
# OBSERVATIONS

- **GP works on this problem**
- **GP determines the size and shape of the solution**
  - **number of operations needed to solve the problem**
  - **size and shape of the program tree**
  - **content of the program tree (i.e., sequence of operations)**
- **GP operates the same whether the solution is linear, polynomial, a rational fraction of polynomials, exponential, trigonometric, etc.**

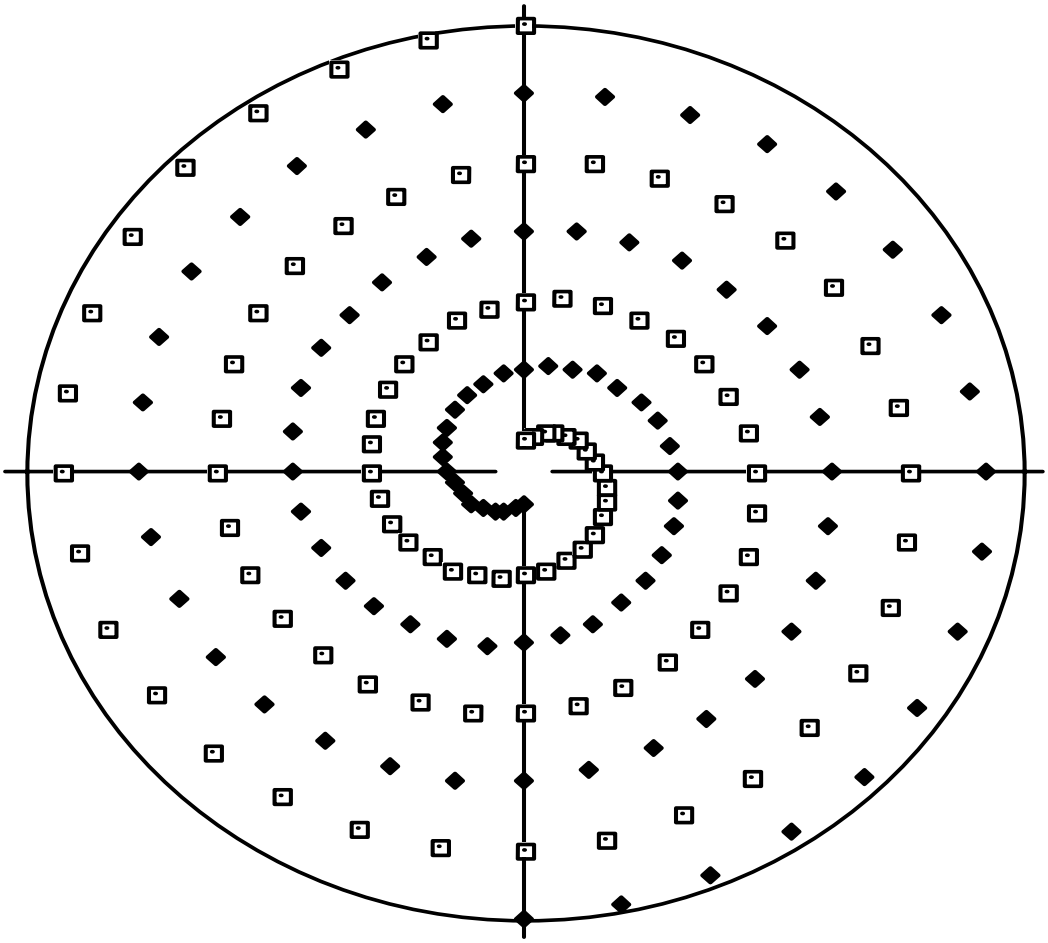
# OBSERVATIONS — CONTINUED

- **It's *not* how a human programmer would have done it**
  - ***Cos (X - X)* = 1**
  - **Not parsimonious**
- **The extraneous functions – `SIN`, `EXP`, `RLOG`, and `RCOS` are absent in the best individual of later generations because they are detrimental**
  - ***Cos (X - X)* = 1 is the exception that proves the rule**
- **The answer is algebraically correct (hence no further cross validation is needed)**

# STRUCTURE ARISES FROM FITNESS



# INTER-TWINED SPIRALS CLASSIFICATION PROBLEM

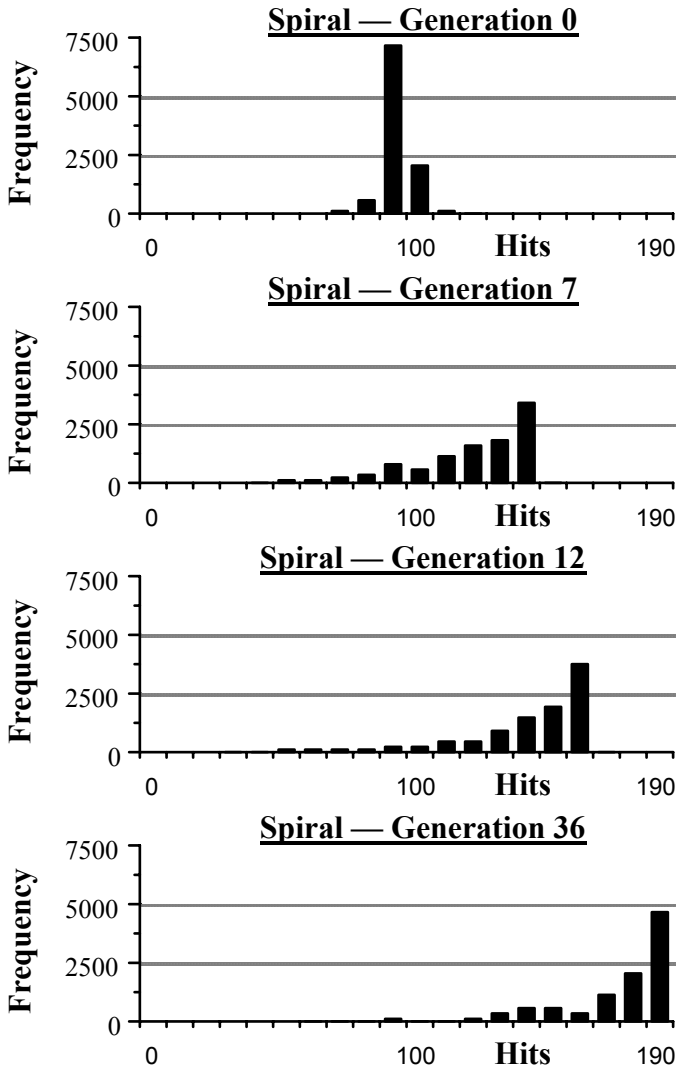


# TABLEAU FOR INTER-TWINED SPIRALS

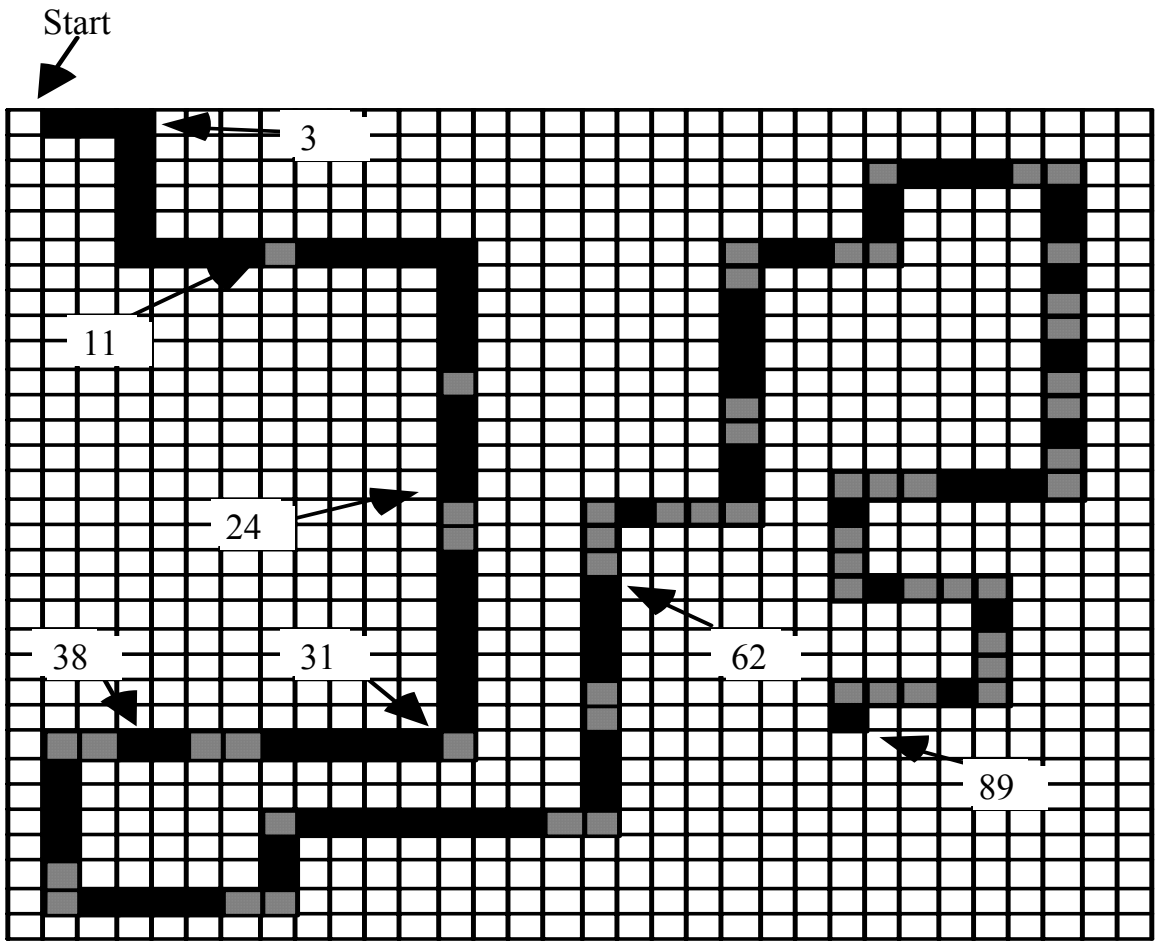
| | |
|---|---|
| **Objective:** | **Find a way to tell whether a given point in the *x*-*y* plane belongs to the first or second of two intertwining spirals.** |
| **Terminal set:** | **$_X$, $_Y$, ℜ, where ℜ is the ephemeral random floating-point constant ranging between –1.000 and +1.000.** |
| **Function set:** | **`+, -, *, %, IFLTE, SIN, COS`.** |
| **Fitness cases:** | **194 points in the *x*-*y* plane.** |
| **Raw fitness:** | **The number of points that are correctly classified.** |
| **Standardized fitness:** | **The maximum raw fitness (i.e., 194) minus the raw fitness.** |
| **Hits:** | **Equals raw fitness for this problem.** |

| | |
|---|---|
| **Wrapper:** | **Maps any S-expression returning a positive value to class +1 and maps all other values to class –1.** |
| **Parameters:** | ***M*** **= 10,000 (with over-selection).** ***G*** **= 51.** |
| **Success predicate:** | **An S-expression scores 194 hits.** |

# IINTER-TWINED SPIRALS HITS HISTOGRAMS



# ARTIFICIAL ANT PROBLEM
# SANTA FE TRAIL



# 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.** |

# TYPICAL PROGRAMS OF GENERATION 0 – ARTIFICIAL ANT

- **Fails to move**

```
(PROGN2 (RIGHT) (LEFT))
```

- **Moves without looking or turning (i.e., doesn't use sensory information)**

```
(PROGN2 (MOVE) (MOVE))
```

- **Contains conditional, but doesn't move**

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

- **Contains conditional, but doesn't info**

```
(IF-FOOD-AHEAD (RIGHT) (RIGHT))
```

- **Contains conditional, but uses the information imprudently**

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

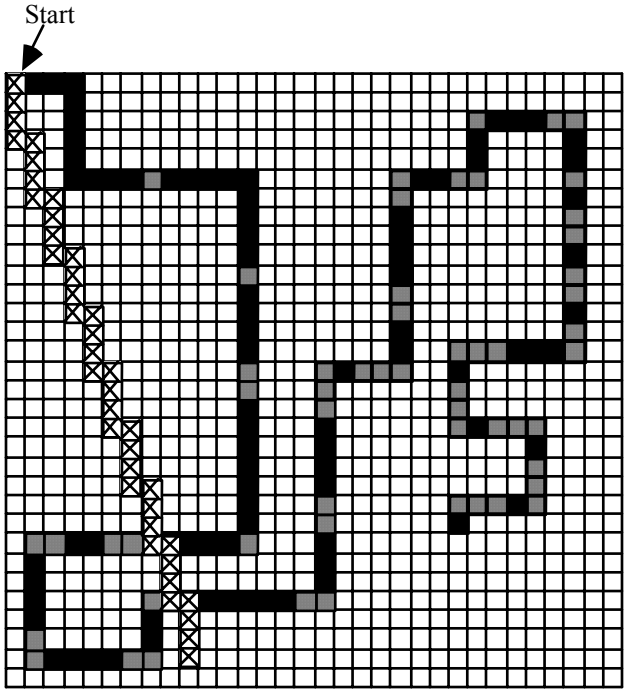
# GENERATION 0 – ARTIFICIAL ANT

- **Average fitness of generation 0 is 3.5 (out of 89)**

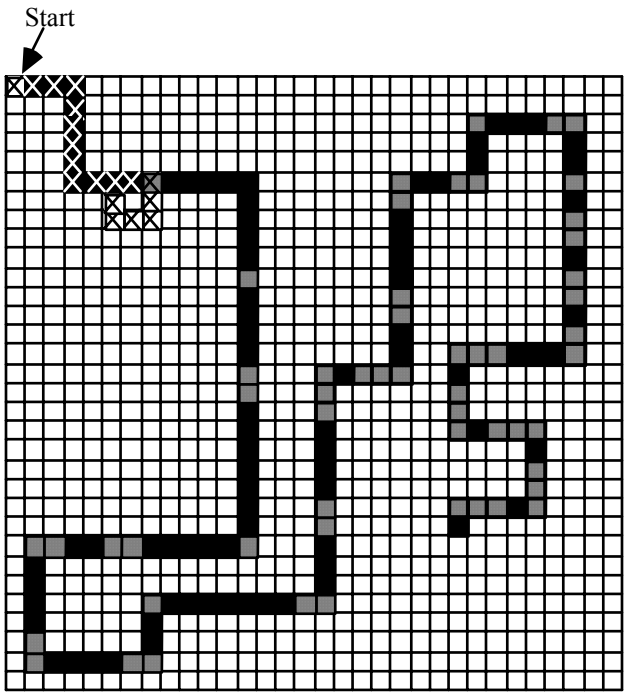
- **Fitness of best-of-generation individual in generation 0 is 32 (out of 89)**

# "QUILTER" – GENERATION 0 – ARTIFICIAL ANT

```
(PROGN3 (RIGHT)
    (PROGN3 (MOVE) (MOVE)
(MOVE))
    (PROGN2 (LEFT) (MOVE)))
```

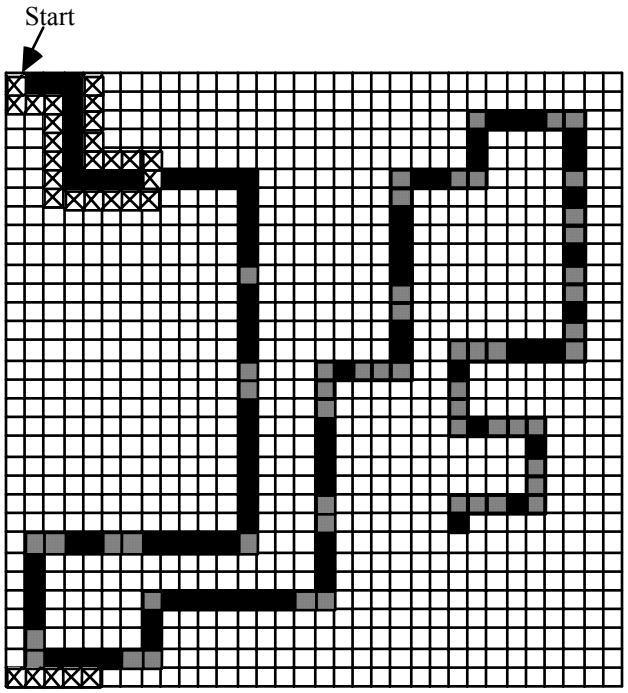


# "LOOPER" – GENERATION 0 – ARTIFICIAL ANT

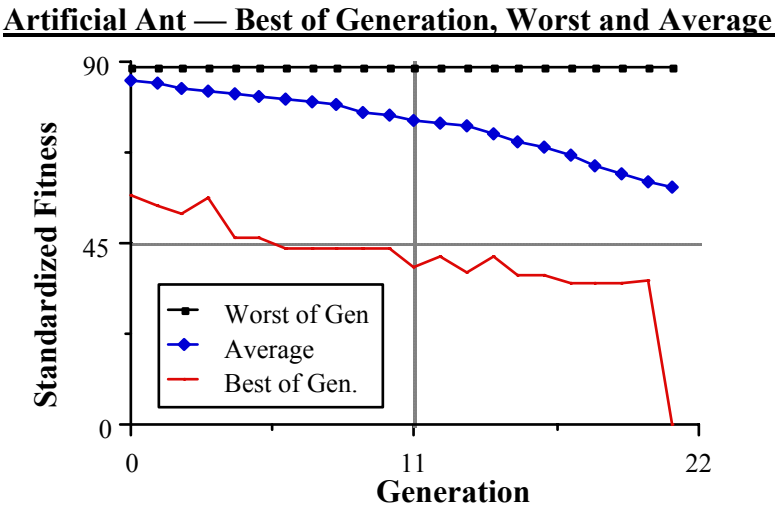


# "AVOIDER" – GENERATION 0 – ARTIFICIAL ANT

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

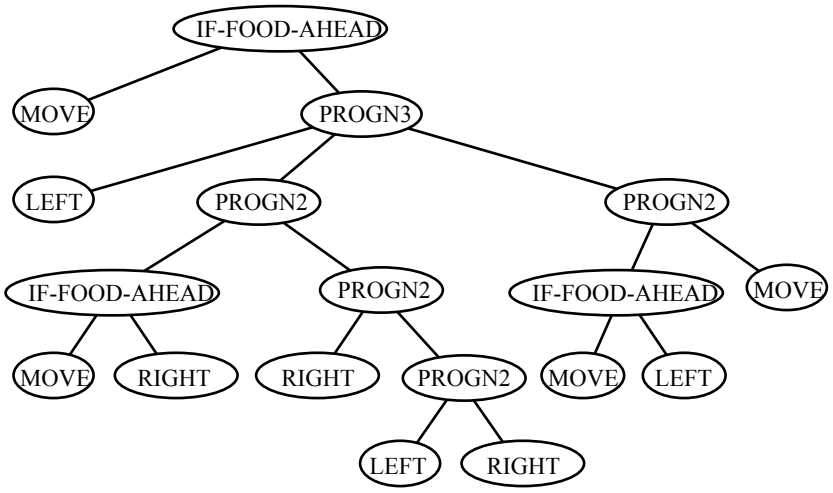


# FITNESS CURVES – ARTIFICIAL ANT – SANTA FE TRAIL



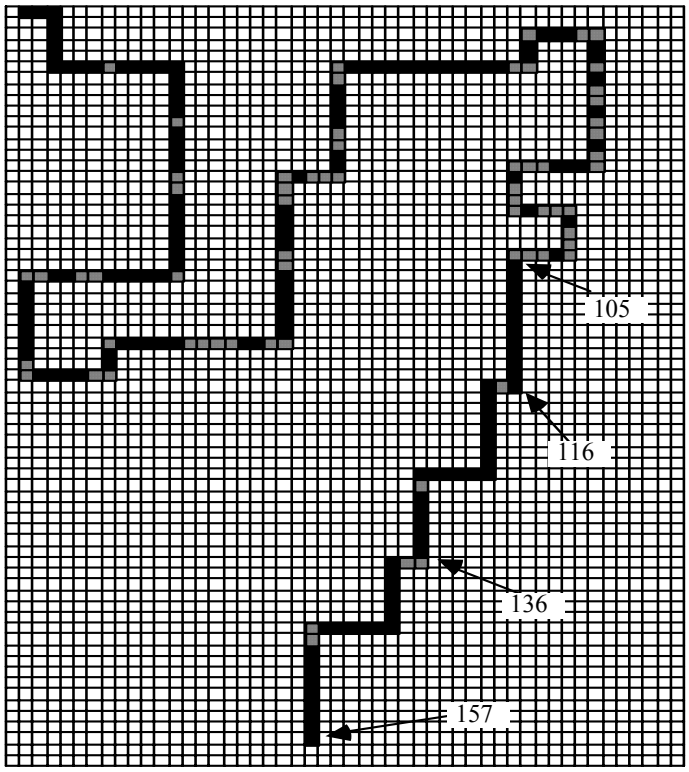
# BEST-OF-RUN INDIVIDUAL – GENERATION 21 – ARTIFICIAL ANT

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



# ARTIFICIAL ANT – LOS ALTOS HILLS TRAIL

- **Pieces of food is 157 (formerly 89)**
- **Increase the number of time steps to 3,000**
- **Increase population size, *M*, to 2,000**

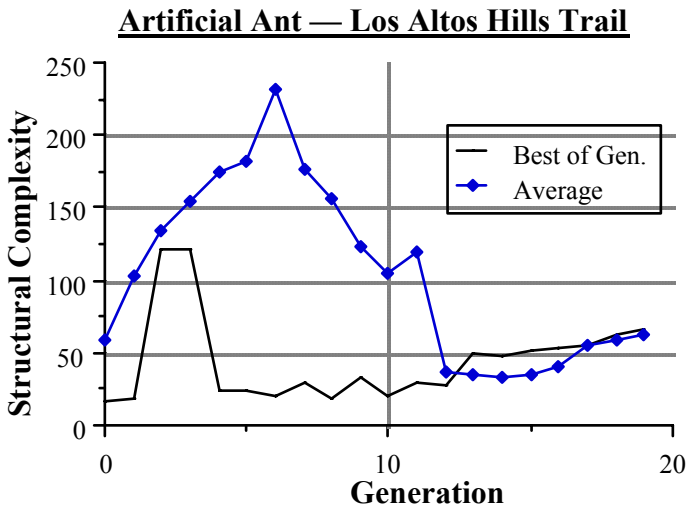


# BEST-OF-RUN INDIVIDUAL – GENERATION 19 – ARTIFICIAL ANT – LOS ALTOS HILLS TRAIL

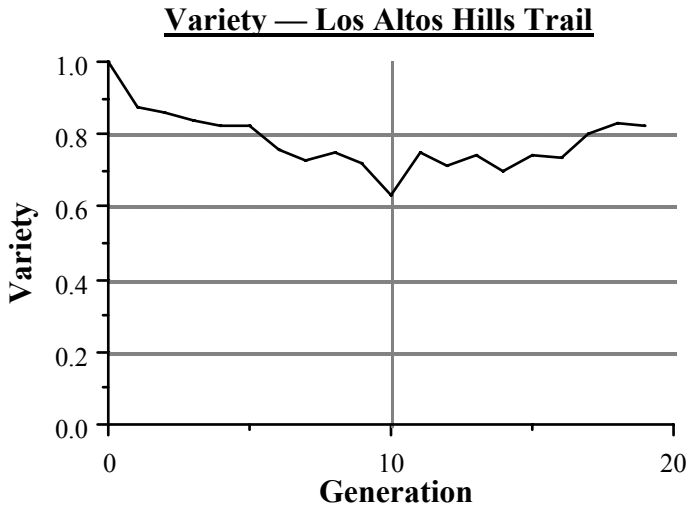
- **Finds all 157 pieces of food in 1,808 steps**

```
(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))
```

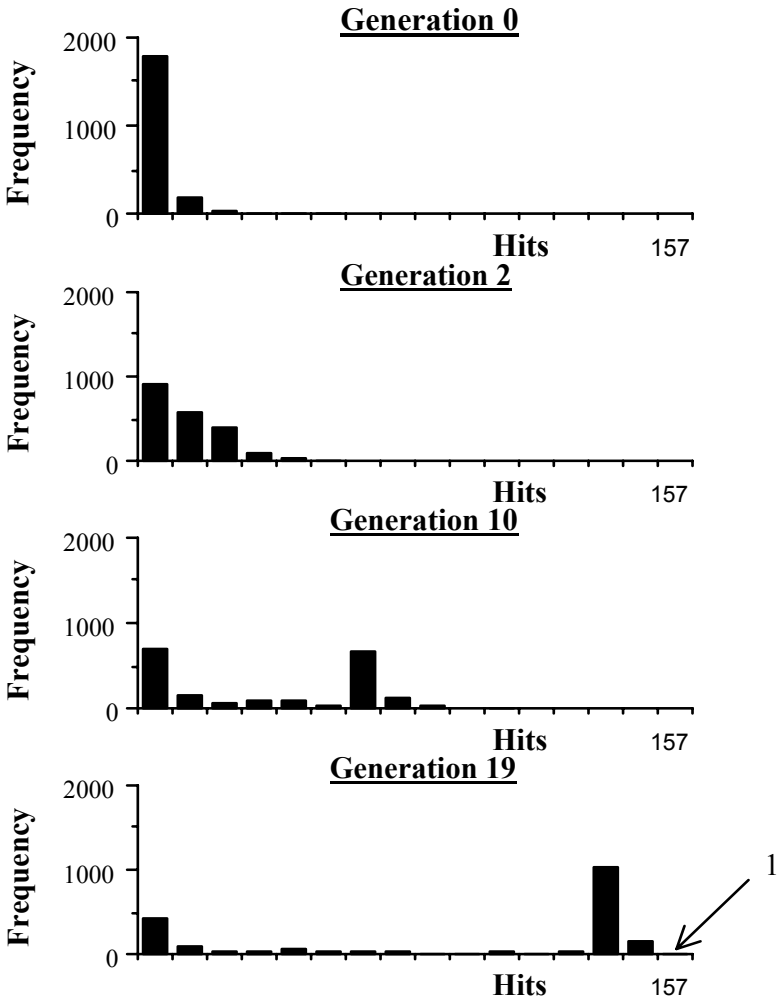
# STRUCTURAL COMPLEXITY CURVES – ARTIFICIAL ANT – LOS ALTOS HILLS TRAIL



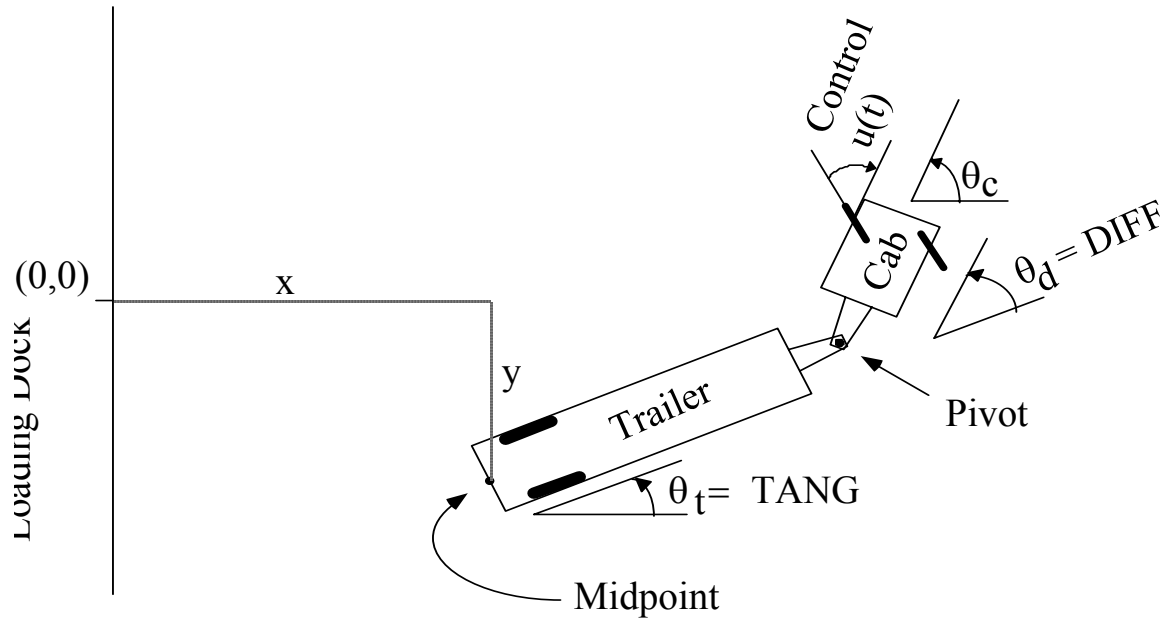
# VARIETY CURVES – ARTIFICIAL ANT – LOS ALTOS HILLS TRAIL



# HITS HISTOGRAMS – ARTIFICIAL ANT – LOS ALTOS HILLS TRAIL



# TRUCK BACKER UPPER (FOUR DIMENSIONAL CONTROL PROBLEM)



# TRUCK BACKER UPPER

- **4-Dimensional state space**
  - **horizontal position, *x***
  - **vertical position, *y***
  - **angle between trailer and horizontal, $\Theta_t$**
  - **angle between trailer and cab, $\Theta_d$**
- **One control variable (angle)**
- **State transition equations map the current state (the 4 variables) into one output (control variable ).**

# TRUCK BACKER UPPER

$$A = \mathbf{r}\cos u[\mathbf{t}]$$

$$B = A\cos(\theta_c[t] - \theta_t[t])$$

$$C = A\sin(\theta_c[\mathbf{t}] - \theta_t[\mathbf{t}])$$

$$\mathbf{x}[t+1\} = \mathbf{x}[t] - B\cos\theta_t$$

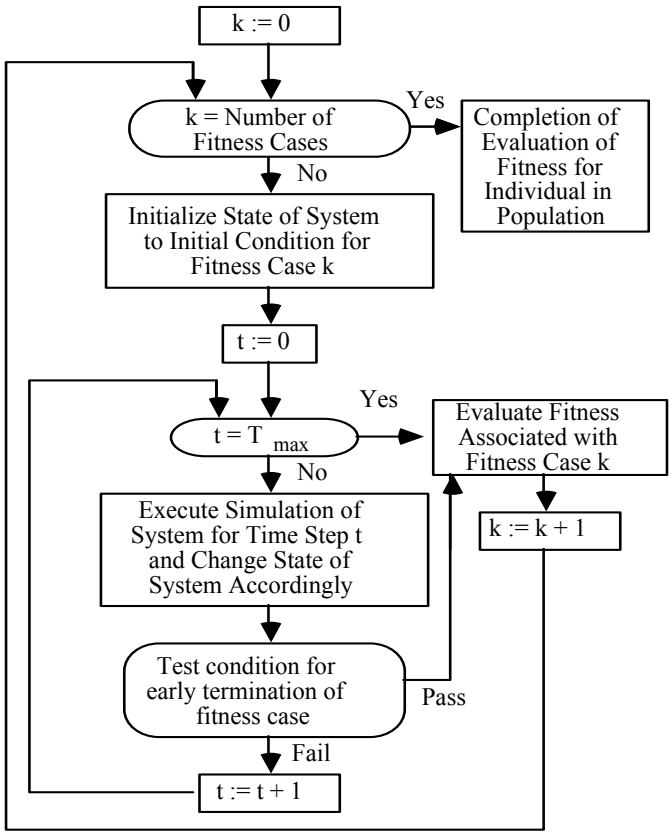
$$\mathbf{y}[t+1] = \mathbf{y}[t] - B\sin\theta_t$$

$$\theta_c[t+1] = \tan^{-1}\left(\frac{d_C\sin\theta_C[t] - r\cos\theta_C[t]\sin u[t]}{d_C\cos\theta_C[t] + r\sin\theta_C[t]\sin u[t]}\right)$$

$$\theta_t[t+1] = \tan^{-1}\text{Error!})$$

$$\theta_d[t] = \theta_t[t] - \theta_c[t]$$

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

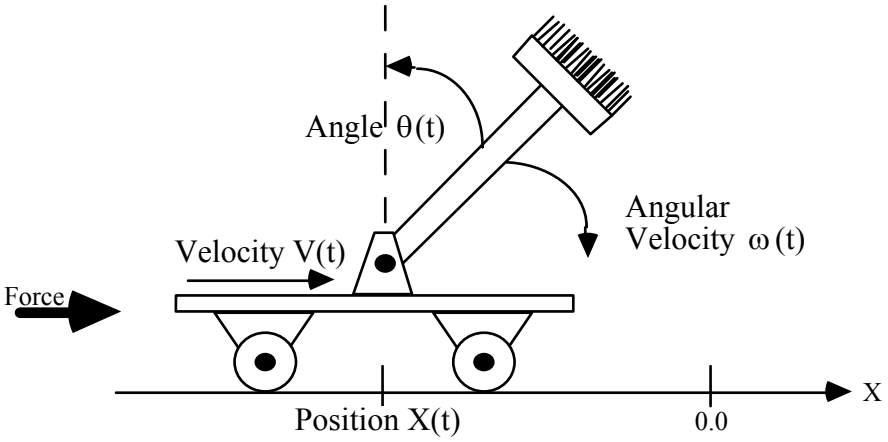


# TABLEAU FOR TRUCK BACKER UPPER

| | |
|---|---|
| **Objective:** | **Find a control strategy for backing up a tractor-trailer truck to a loading dock.** |
| **Terminal set:** | **X, Y, TANG, DIFF, and the ephemeral random constant ℜ ranging from –1.000 to +1.000.** |
| **Function set:** | **+, -, *, %, ATG, IFLTZ** |
| **Fitness cases:** | **8 initial condition points over the state variables X, Y, and TANG (with DIFF of 0).** |
| **Raw fitness:** | **The sum, taken over the 8 fitness cases, of the sum of squares of the difference between the actual values of X, Y, and TANG from their target values.** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |

| | |
|---|---|
| **Hits:** | **Number of fitness cases for which $x$ is less than 0.1 meters, the absolute value of $y$ is less than 0.42 meters, and the absolute value of $\text{TANG}$ is less than 0.25 radians** |
| **Wrapper:** | **Produces a saturated force between –1 radians and +1 radians.** |
| **Parameters:** | ***M*** **= 1,000 (with over-selection).** ***G*** **= 51.** |
| **Success predicate:** | **An S-expression scores 8 hits.** |

# BROOM BALANCING



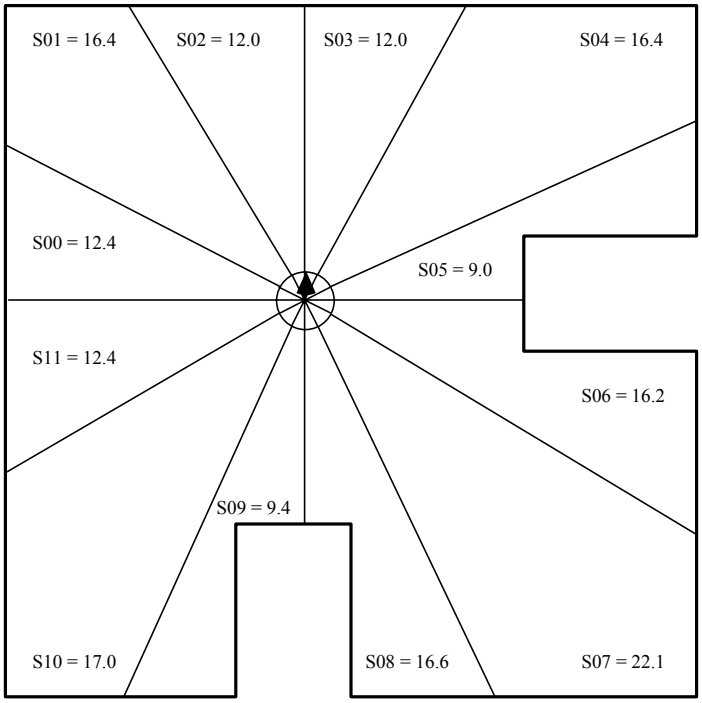
# TABLEAU FOR BROOM BALANCING

| | |
|---|---|
| **Objective:** | **Find a control strategy to balance the broom and bring the cart to rest in minimal time.** |
| **Terminal set:** | **VEL (velocity *v*), ANG (angle θ), AVL (angular velocity ω), and the ephemeral random floating-point constant ℜ ranging from –1.000 to +1.000.** |
| **Function set:** | **+, -, *, %, SIG, ABS, SRT, SQ, CUB, GT.** |
| **Fitness cases:** | **10 initial condition points in state space of the problem (*v*, θ,ω).** |
| **Raw fitness:** | **The sum, over the fitness cases, of the times required to balance the broom and bring the cart to rest.** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |
| **Hits:** | **Number of fitness cases that do 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.** |
| **Parameters:** | ***M* = 500. *G* = 51.** |
| **Success predicate:** | **None.** |

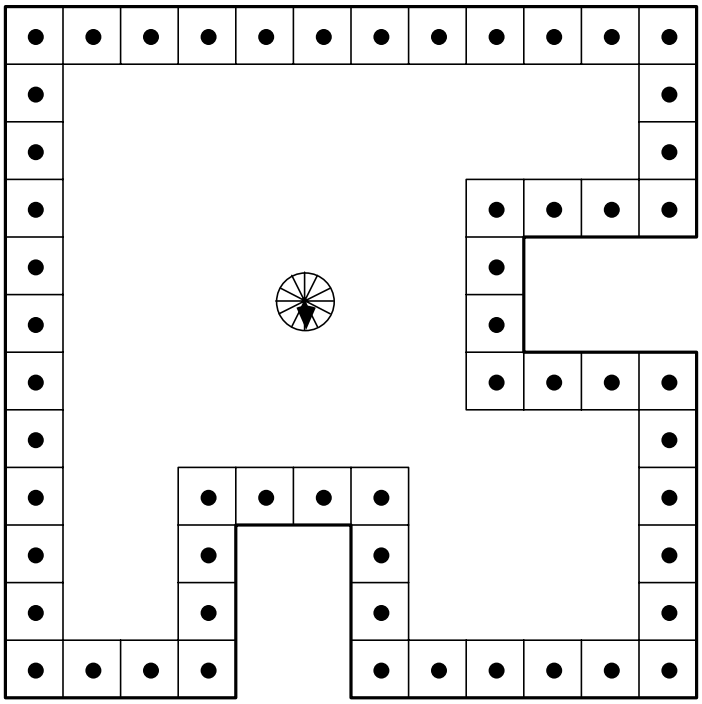
# WALL-FOLLOWING PROBLEM

# 12 SONAR SENSORS



# WALL-FOLLOWING PROBLEM

# FITNESS MEASURE



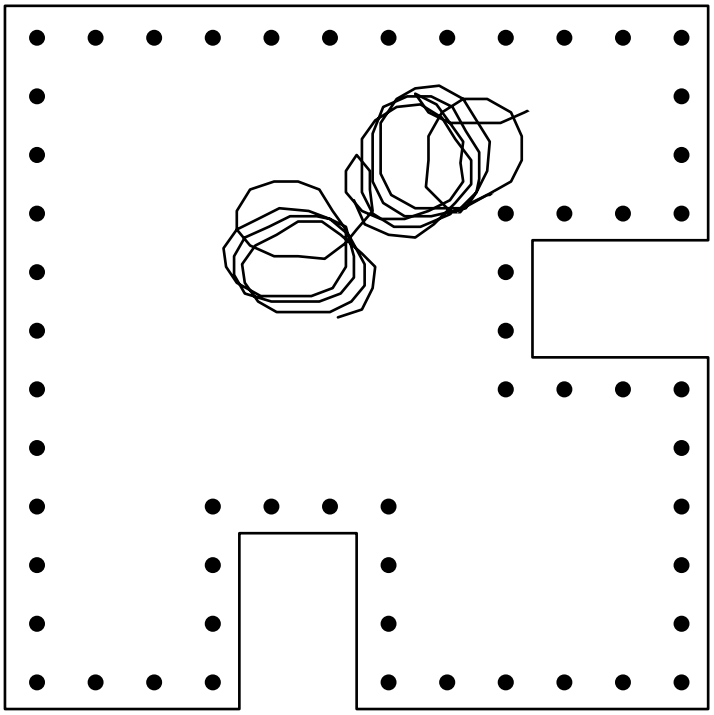
# TABLEAU FOR THE WALL-FOLLOWING PROBLEM

| | |
|---|---|
| **Objective:** | **Find a computer program which, when executed by a mobile robot moves along the periphery of an irregular room.** |
| **Terminal set:** | **The distances to the nearest wall in each of 12 directions as reported by 12 sonar senses, two constants MSD and EDG, one derived value SS, Move Forward (MF), Move Backwards (MB), Turn Right (TR), and Turn Left (TL).** |
| **Function set:** | **If-Less-Than-Or-Equal (IFLTE), and the connective PROGN2.** |
| **Fitness cases:** | **One fitness case.** |
| **Raw fitness:** | **Number of 2.3 foot square tiles along the periphery of the room that are touched by the robot within an allotted time of 400 time steps.** |

| | |
|---|---|
| **Standardized fitness:** | **Total number of tiles (56) minus raw fitness.** |
| **Hits:** | **Equals raw fitness for this problem.** |
| **Wrapper:** | **None.** |
| **Parameters:** | ***M* = 1,000 (with over-selection). *G* = 51 (see text).** |
| **Success predicate:** | **An S-expression scores 56 hits.** |

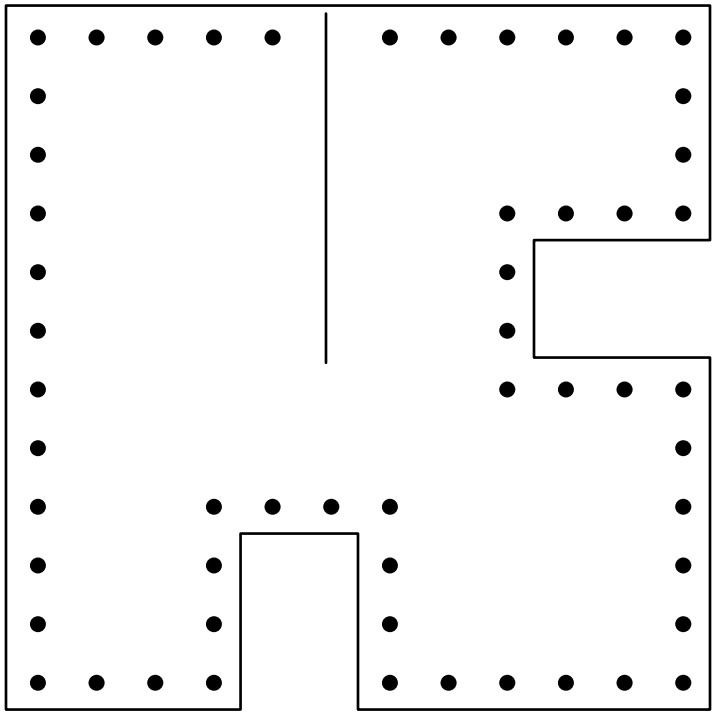
# WALL-FOLLOWING PROBLEM

# AIMLESS WANDERING TRAJECTORY OF ONE PROGRAM (OUT OF 1,000) OF GENERATION 0



# WALL-FOLLOWING PROBLEM

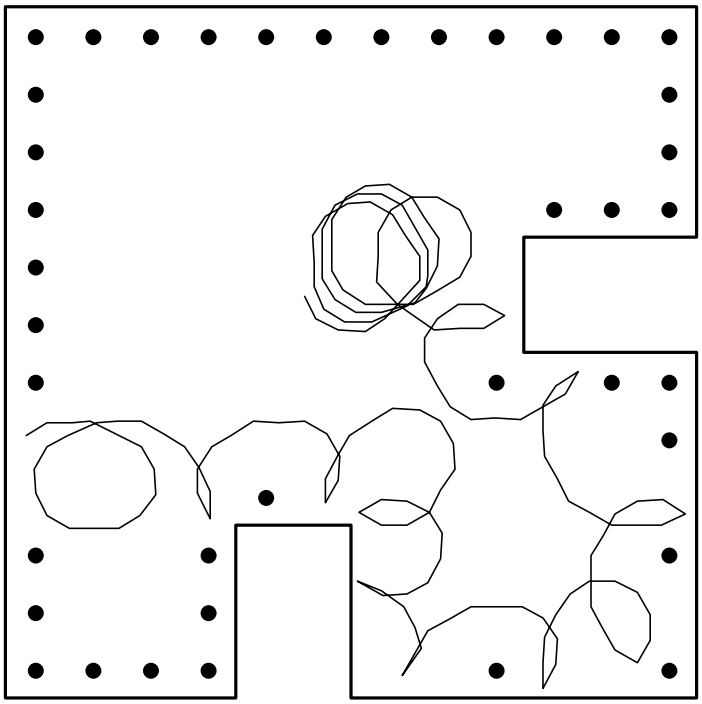
# WALL-BANGING TRAJECTORY OF ONE PROGRAM (OUT OF 1,000) OF GENERATION 0



# WALL-FOLLOWING PROBLEM
# BEST-OF-GENERATION PROGRAM
# (OUT OF 1,000) OF GENERATION 0

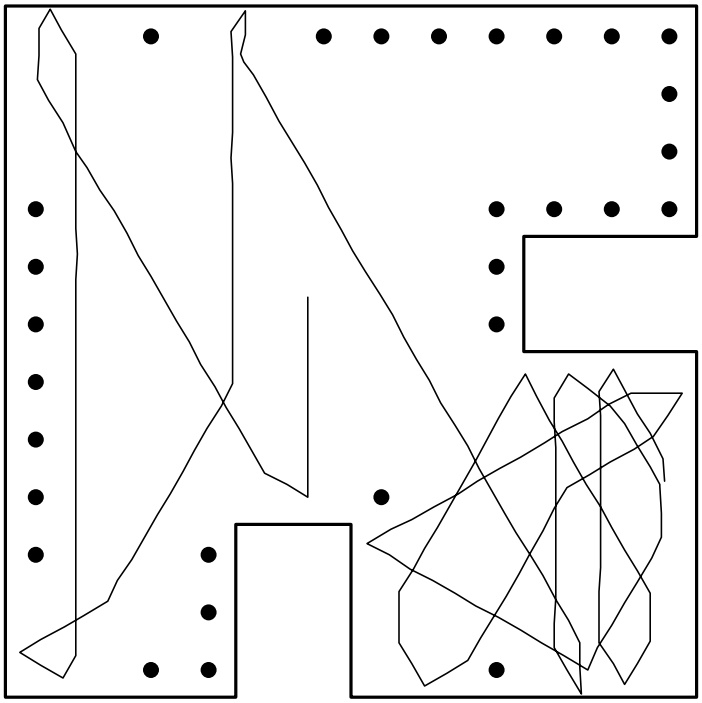
- **Scored 17 (out of 56). Has 17 points.**

```
(IFLTE (PROGN2 MSD (TL))
       (IFLTE S06 S03 EDG (MF))
       (IFLTE MSD EDG S05 S06)
       (PROGN2 MSD (MF)))
```



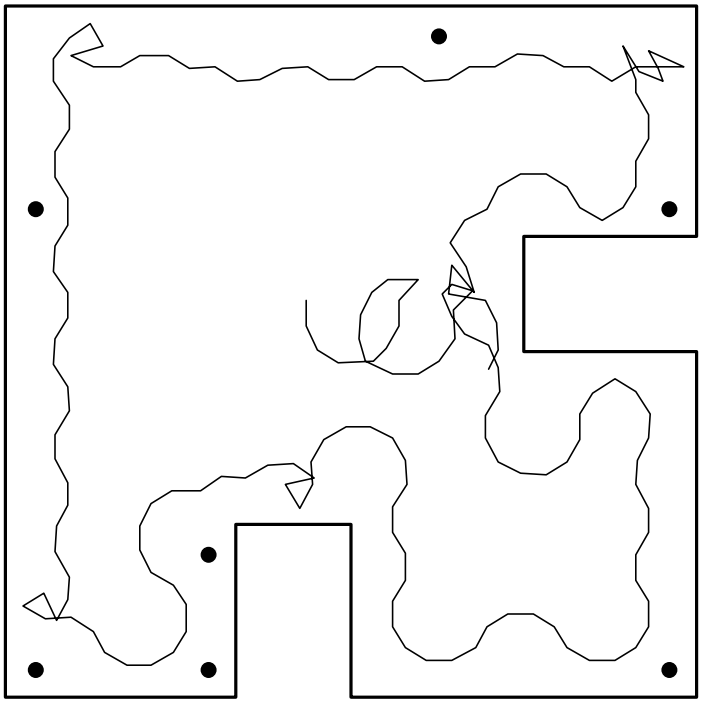
# WALL-FOLLOWING PROBLEM
# BEST PROGRAM OF GENERATION 2

- **Scores 27 (out of 56). Has 57 points.**



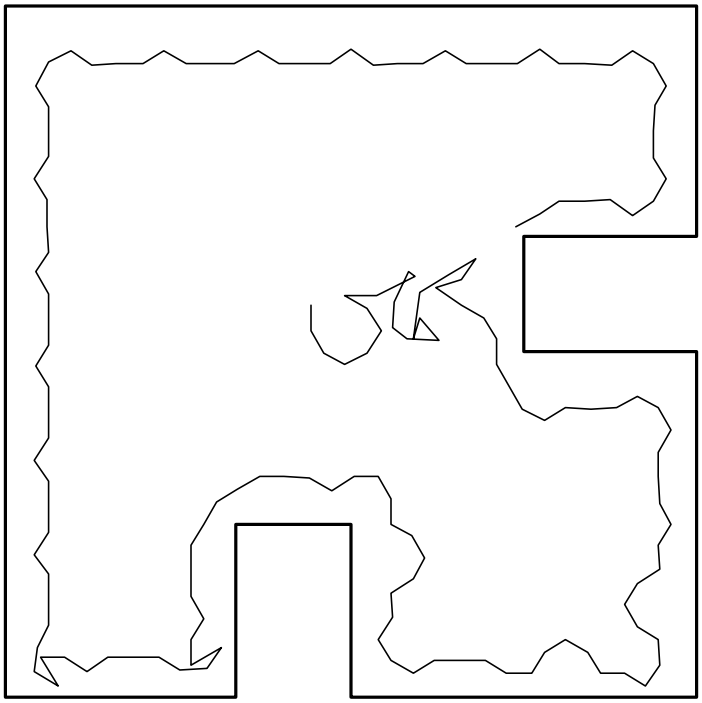
# WALL-FOLLOWING PROBLEM
# BEST PROGRAM OF GENERATION 14

- **Scores 49 (out of 56). Has 45 points.**



# WALL-FOLLOWING PROBLEM
# BEST PROGRAM OF GENERATION 57

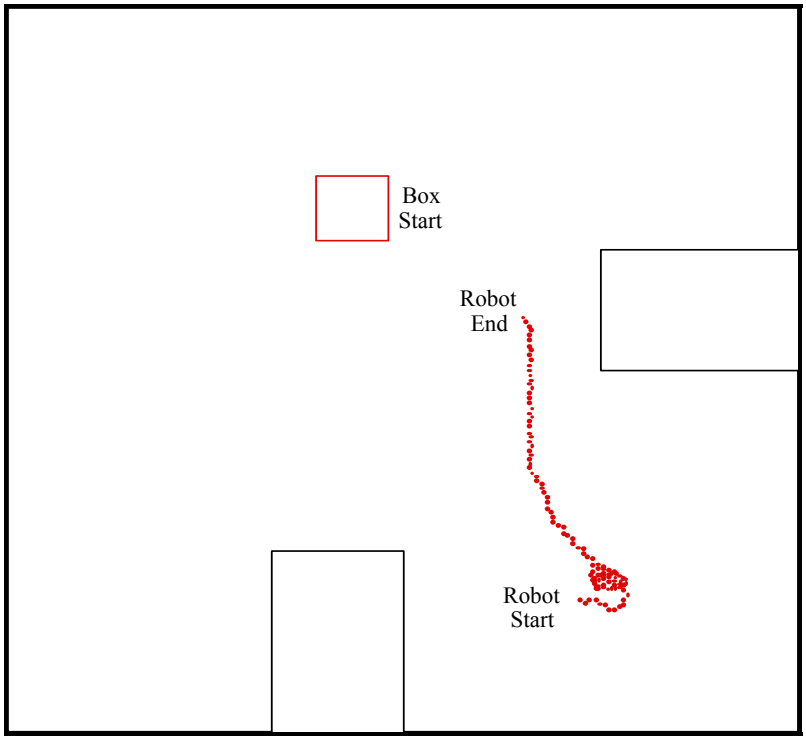
- **Scores 56 (out of 56). Has 145 points.**



# BOX MOVING PROBLEM

- Same room, same functions and terminals, same population size
- Different fitness measure.
  - **4 initial conditions (starting positions for robot)**
  - **Fitness is sum, over the 4 fitness cases, of the distance between the box and the wall**

## TRAJECTORY OF ROBOT – GEN 0

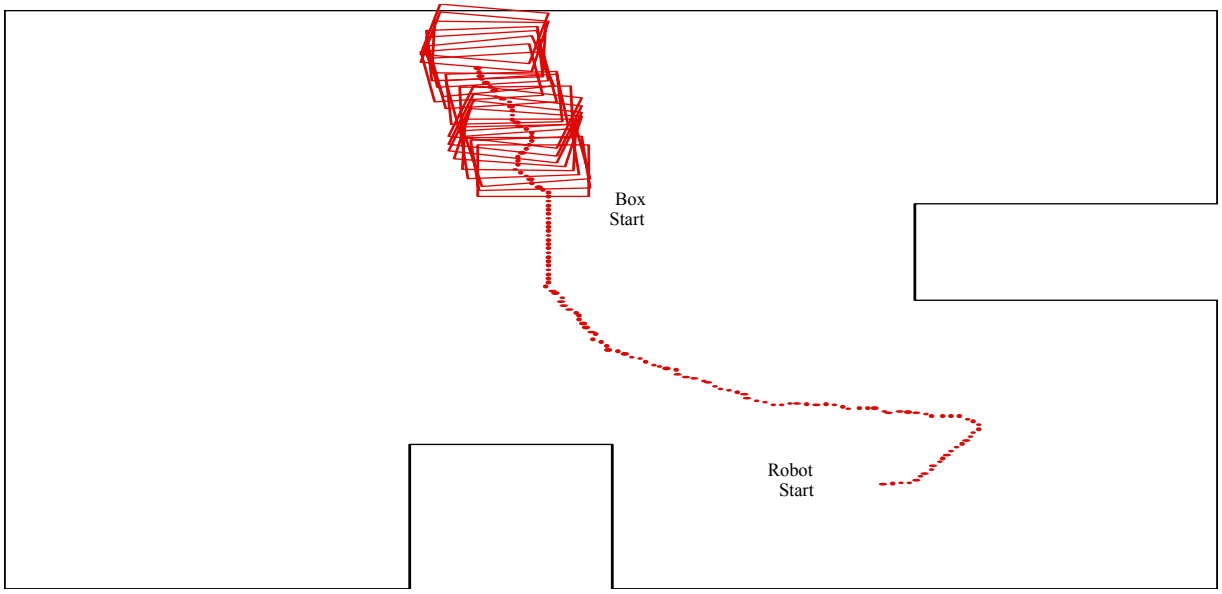


# TABLEAU FOR BOX MOVING PROBLEM

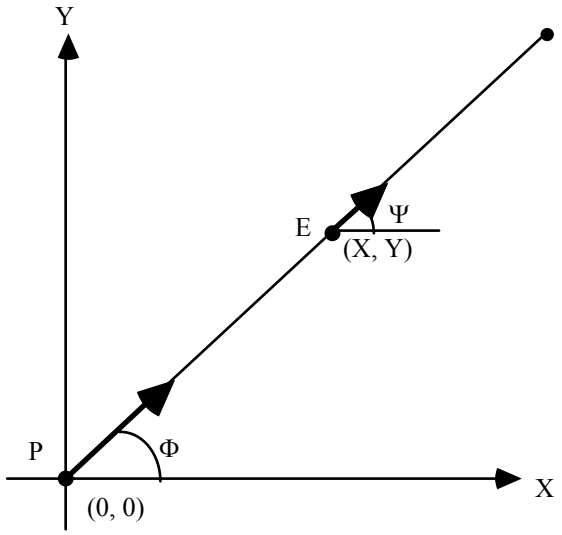
| | |
|---|---|
| **Objective:** | **Find a computer program which, when executed by a mobile robot, finds a box in an irregular room and moves it to a wall.** |
| **Terminal set:** | **The distances to the nearest object in each of 12 directions relative to the current facing direction of the robot as reported by 12 sonar sensors, one derived value SS, Move Forward (MF), Turn Right (TR), Turn Left (TL).** |
| **Function set:** | **If-Less-Than-Or-Equal (IFLTE), the connective PROGN2, IFBMP (If Bumped), and IFSTK (If Stuck).** |
| **Fitness cases:** | **Four fitness cases representing four starting positions of the robot in various parts of the room.** |

| | |
|---|---|
| **Raw fitness:** | **The sum, over the four fitness cases, of the distance between the wall and the point on the box that is closest the nearest wall. Each fitness case times out when any part of the box touches a wall or after 350 time steps.** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |
| **Hits:** | **Number of fitness cases for which the box touches a wall prior to the timing out.** |
| **Wrapper:** | **None.** |
| **Parameters:** | ***M* = 500. *G*= 51.** |
| **Success predicate:** | **An S-expression scores 4 hits.** |

# BOX MOVING–GEN 45– SE CASE



# DIFFERENTIAL PURSUER-EVADER GAME



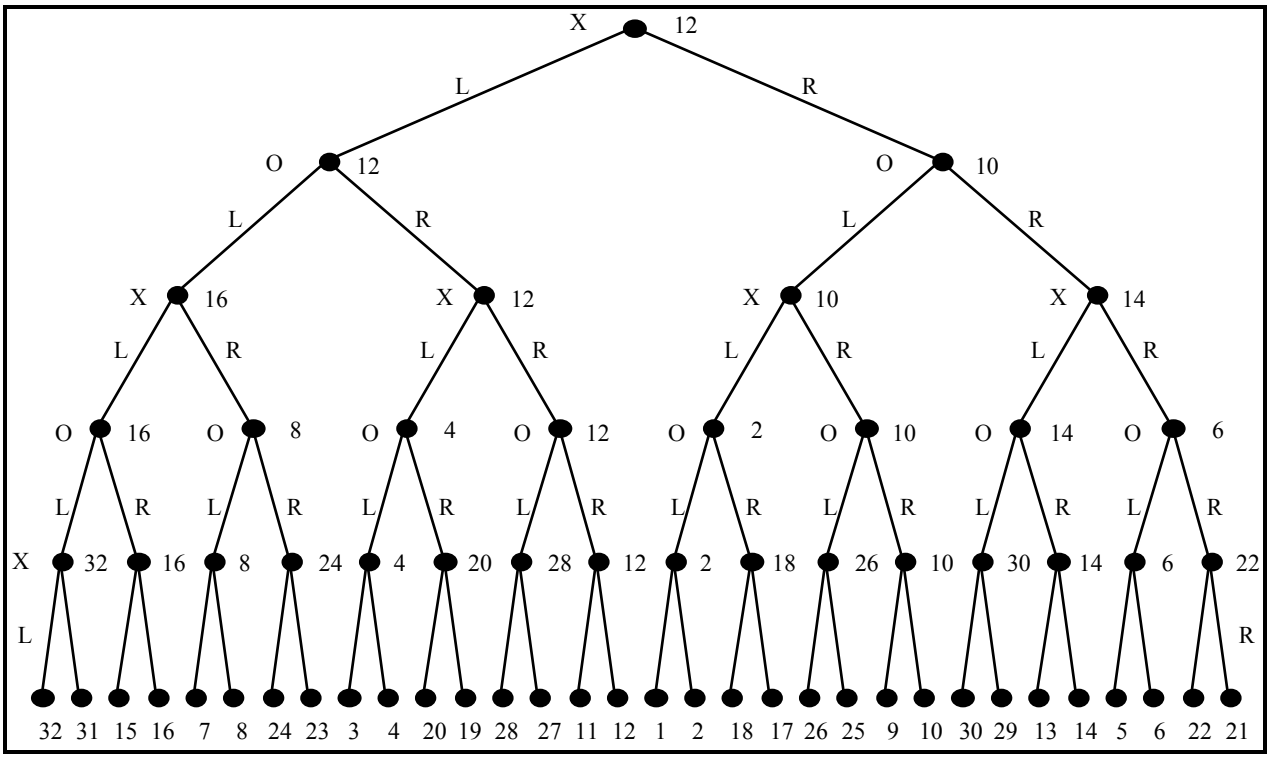

## Pursuer P and Evader E

# TABLEAU FOR THE DIFFERENTIAL PURSUER-EVADER GAME

| | |
|---|---|
| **Objective:** | **Find the minimax strategy for the pursuer in a differential pursuer-evader game of simple pursuit.** |
| **Terminal set:** | **$_X$, $_Y$, $\Re$., where the ephemeral random constant $\Re$ ranges over –1.000 and +1.000.** |
| **Function set:** | **+, -, *, %, EXP, IFLTZ.** |
| **Fitness cases:** | **20 initial conditions consisting of the initial condition points ($x$, $y$) for the Evader.** |
| **Raw fitness:** | **The time required to capture the Evader, averaged over the 20 fitness cases.** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |
| **Hits:** | **Number of fitness cases that do not time out.** |
| **Wrapper:** | **None.** |
| **Parameters:** | **$M$ = 500. $G$ = 51.** |

| | |
|---|---|
| **Success predicate:** | **None.** |

# TABLEAU FOR MINIMAX STRATEGY FOR DISCRETE 32-OUTCOME GAME



# TABLEAU FOR MINIMAX STRATEGY FOR DISCRETE 32-OUTCOME GAME

| | |
|---|---|
| **Objective:** | **Find the minimax strategy for playing a discrete 32-outcome game in extensive form for player X.** |
| **Terminal set:** | `L, R.` |
| **Function set:** | `CXM1, COM1, CXM2, COM2.` |
| **Fitness cases:** | **For player X, the fitness cases consist of the four possible combinations of O moves, namely O choosing `L` or `R` for moves 1 and 2.** |
| **Raw fitness:** | **The sum, over the four fitness cases, of the payoffs to player X.** |
| **Standardized fitness:** | **The maximum sum (i.e., 32 times 4) minus raw fitness.** |
| **Hits:** | **Number of fitness cases for which player X receives a payoff of at least as good as the minimax strategy.** |
| **Wrapper:** | **None.** |

| **Parameters:** | ***M* = 500. *G* = 51.** |
|---|---|
| **Success predicate:** | **An S-expression scores 4 hits.** |

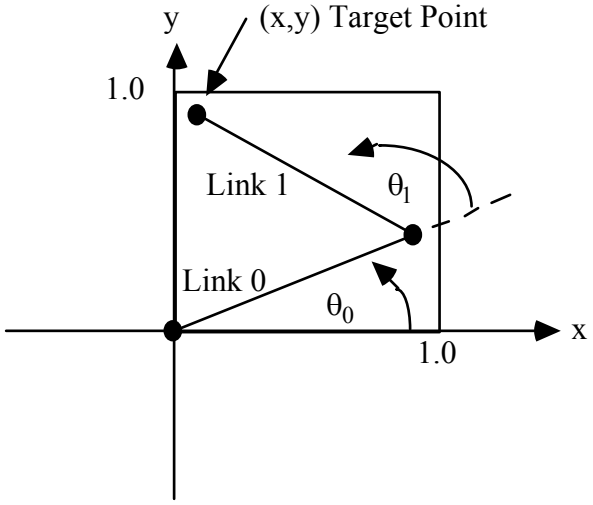
# TABLEAU FOR THE PROBLEM OF CO-EVOLVING STRATEGIES FOR THE DISCRETE 32-OUTCOME GAME

| | |
|---|---|
| **Objective:** | **Simultaneously co-evolve the minimax strategy for playing a discrete 32-outcome game in extensive form for both player X and player O.** |
| **Terminal set:** | `L, R.` |
| **Function set:** | `CXM1, COM1, CXM2, COM2.` |
| **Fitness cases:** | **All 300 individuals in the opposing population.** |
| **Raw fitness:** | **The average, over fitness cases consisting of all the individuals in the opposing population, of the payoffs to the individual strategy.** |
| **Standardized fitness:** | **The maximum possible value of raw fitness minus the raw fitness of the individual.** |

| | |
|---|---|
| **Hits:** | **Number of fitness cases for which the payoff to an individual equals or exceeds the value of the game.** |
| **Wrapper:** | **None.** |
| **Parameters:** | ***M* = 300. *G* = 51.** |
| **Success predicate:** | **None.** |

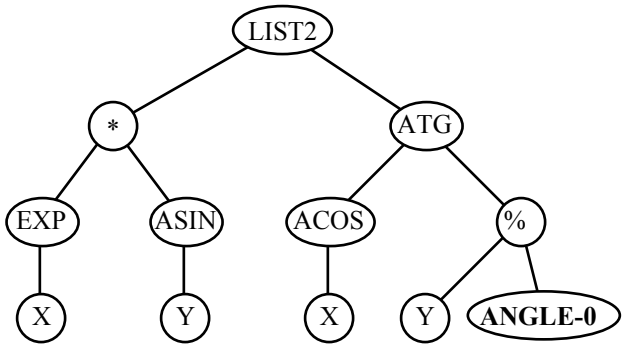
# INVERSE KINEMATICS

**Robot arm consisting of link 0 at angle $\theta_0$ from the *x* axis and link 1 at angle $\theta_1$ from link 0 so that the endpoint of link l reaches the target point (*x*,*y*)**



# INVERSE KINEMATICS

## Random individual from generation 0 for the inverse kinematics problem



# INVERSE KINEMATICS

## Best-of-generation individual from generation 154 – 25 hits – 231 points

```
(LIST2

          (% (* (- (% (* (- X Y) Y) (ACOS% (EXP X)))
(ACOS% Y)) (ATG (+ (+ (ACOS% Y) (% (% (- X Y) (ACOS%
(EXP X))) (ACOS% (EXP X)))) (* (* (% (- X Y) (ACOS%
(% (% (* (- X (% Y X)) Y) (ACOS% (EXP X))) (ACOS%
X)))) Y) Y)) (% Y X))) (% (EXP X) (ACOS% (EXP X))))

          (+ (ACOS% X) (ATG (+ (EXP (ATG (ATG (+
(ATG (ACOS% Y) Y) (* (ATG (+ (ATG (+ (ACOS% X) ANGLE-
0) (* X (ACOS% X))) (ACOS% X)) Y) (ATG X Y))) (ACOS%
X)) (ATG (ATG (ATG (ACOS% X) (* (ATG (+ (* X (ATG X
Y)) (+ (ACOS% Y) ANGLE-0)) (+ (ACOS% X) (* X (ACOS%
X)))) (ATG (+ (* X (ATG X Y)) ANGLE-0) (+ (ACOS% X)
(* (ACOS% X) (ACOS% X)))))) (* (ATG (+ (ACOS% X) (ATG
(+ (ACOS% X) ANGLE-0) (ACOS% X))) (+ (ACOS% X) (* X
(ACOS% X)))) (ATG X (* X (+ (ACOS% X) (ACOS% Y))))))
(* (ATG (+ (+ (ATG (+ (ACOS% X) ANGLE-0) (ACOS% X))
(ACOS% X)) (+ (ATG (+ (ACOS% X) ANGLE-0) (ACOS% X))
(ACOS% X))) (+ (ATG X Y) (* X (+ (ACOS% X) (ACOS%
X))))) (* (ATG (+ (ATG (+ (ATG Y X) ANGLE-0) (ACOS%
Y)) (ACOS% X)) (ATG Y X)) (ATG X Y)))))) (* X X))
(ATG (+ (ACOS% X) ANGLE-0) (% (ACOS% Y) (+ Y X))))))
```

# TABLEAU FOR INVERSE KINEMATICS

| | |
|---|---|
| **Objective:** | **Find vector of two angles to move a two-link robot arm to a target point (*x*,*y*) in the plane.** |
| **Terminal set:** | • **For the first component: T0 = {X,Y,ℜ}.**<br>• **For the second component: T1 = {ANGLE-0, X, Y, ℜ}.** |
| **Function set:** | LIST2, +, -, *, %, EXP, ASIN, ACOS, ATG. |
| **Fitness cases:** | **25 target points (x,y) in unit square, either chosen at random or in a regular 5 ∞ 5 grid arrangement.** |
| **Raw fitness:** | **The sum, taken over the 25 fitness cases, of the distances between the endpoint of the second link of the robot arm and the particular target point.** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |

| | |
|---|---|
| **Hits:** | **Endpoint of the second robot arm is within 0.05 of the target point.** |
| **Wrapper:** | **None.** |
| **Parameters:** | ***M* = 500. *G* = 51.** |
| **Success predicate:** | **An S-expression scores 25 hits.** |
| **Rules of construction:** | • **Root is always the** LIST2 **function.**<br>• **First component of the vector is a composition of functions (other than** LIST2**) and the terminals** X,Y**,** $\Re$**.**<br>• **Second component of the vector is a composition of functions (other than** LIST2**), the terminal** ANGLE-0**, and the terminals** X**,** Y**, and** $\Re$**.** |

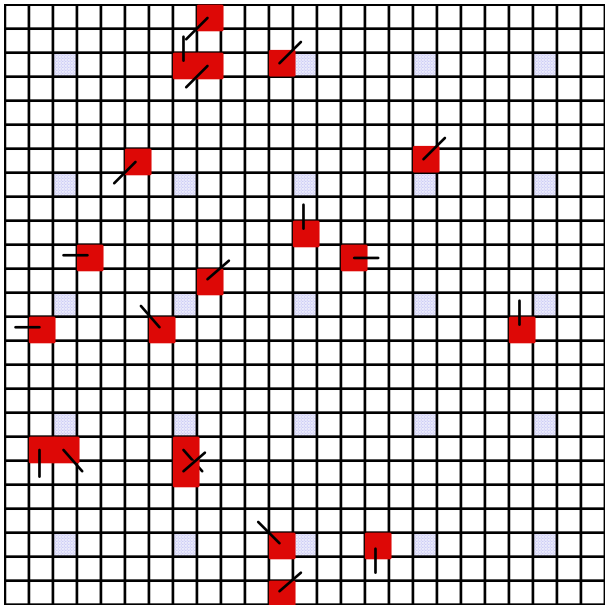
| **Types of points:** | **Three types of points:**<br>• **the root,**<br>• **points in the first component (which do not include ANGLE-0), and**<br>• **points in the second component (which may include ANGLE-0).** |
|---|---|

# TABLEAU FOR EMERGENT COLLECTING BEHAVIOR

| | |
|---|---|
| **Objective:** | **Collect the available food and consolidate it into one compact location.** |
| **Terminal set:** | `(MOVE-RANDOM)`, `(PICK-UP)`, `(MOVE)`, `(DROP-FOOD)`. |
| **Function set:** | `IF-FOOD-HERE`, `IF-CARRYING-FOOD`, `IF-FOOD-ADJACENT`, `PROGN2`. |
| **Fitness cases:** | **One fitness case consisting of the initial positions of the food pellets and agents.** |
| **Raw fitness:** | **The sum, over each of the 25 food pellets, of the distances to each of the other 24 food pellets.** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |
| **Hits:** | **Same as raw fitness for this problem.** |
| **Wrapper:** | **None.** |
| **Parameters:** | $M = 500$. $G = 51$. |
| **Success predicate:** | **None.** |

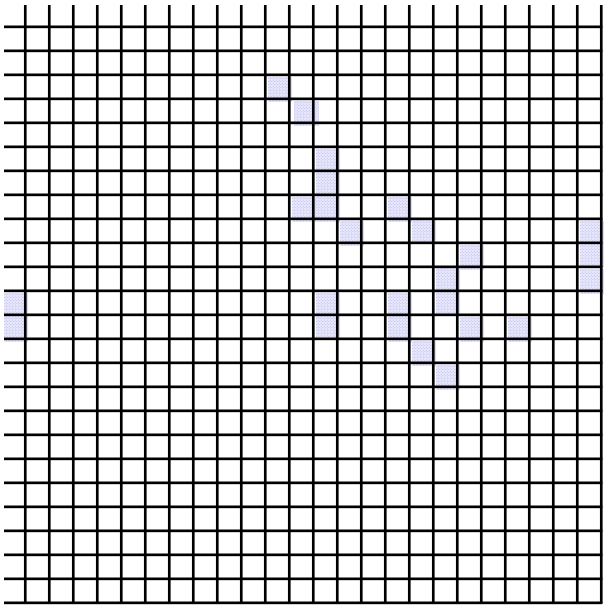
# EMERGENCE OF SORTING BEHAVIOR IN ANTS

## Initial configuration of 25 food pellets and 20 ants for Emergent Sorting problem for ants



# EMERGENCE OF SORTING BEHAVIOR IN ANTS

## Best-of-generation individual from generation 0 – 16 final islands containing the 25 food pellets



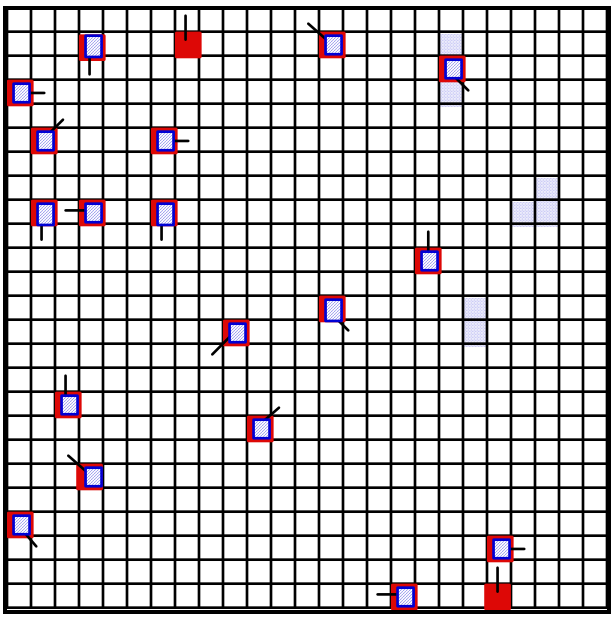
# EMERGENCE OF SORTING BEHAVIOR IN ANTS

## Best-of-generation individual from generation 27 – 69 points

```
(IF-FOOD-ADJACENT (PROGN2 (PROGN2 (PICK-UP) (MOVE))
(IF-FOOD-ADJACENT (IF-FOOD-HERE (PROGN2 (MOVE) (MOVE-
RANDOM)) (IF-FOOD-HERE (MOVE-RANDOM) (DROP-FOOD)))
(PICK-UP))) (IF-FOOD-ADJACENT (IF-CARRYING-FOOD (IF-
CARRYING-FOOD (PROGN2 (IF-CARRYING-FOOD (IF-FOOD-HERE
(IF-FOOD-HERE (IF-CARRYING-FOOD (PICK-UP) (PICK-UP))
(PROGN2 (IF-FOOD-HERE (PROGN2 (PICK-UP) (IF-FOOD-HERE
(PICK-UP) (IF-FOOD-ADJACENT (PICK-UP) (DROP-FOOD))))
(IF-FOOD-ADJACENT (DROP-FOOD) (MOVE))) (MOVE-
RANDOM))) (IF-FOOD-ADJACENT (IF-FOOD-ADJACENT (PICK-
UP) (MOVE)) (IF-CARRYING-FOOD (DROP-FOOD) (IF-
CARRYING-FOOD (PROGN2 (MOVE-RANDOM) (MOVE-RANDOM))
(IF-CARRYING-FOOD (MOVE) (MOVE)))))) (IF-CARRYING-
FOOD (DROP-FOOD) (IF-FOOD-HERE (MOVE-RANDOM) (PROGN2
(PICK-UP) (MOVE))))) (MOVE-RANDOM)) (IF-CARRYING-FOOD
(IF-FOOD-ADJACENT (DROP-FOOD) (DROP-FOOD)) (IF-FOOD-
HERE (DROP-FOOD) (DROP-FOOD)))) (PROGN2 (MOVE-RANDOM)
(MOVE-RANDOM))) (MOVE-RANDOM)))
```

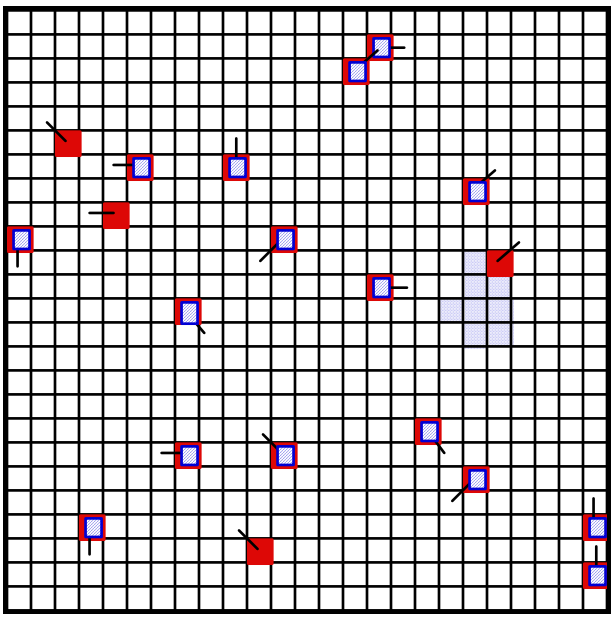
# EMERGENCE OF SORTING BEHAVIOR IN ANTS

## Epoch 400 for best-of-generation individual from generation 27



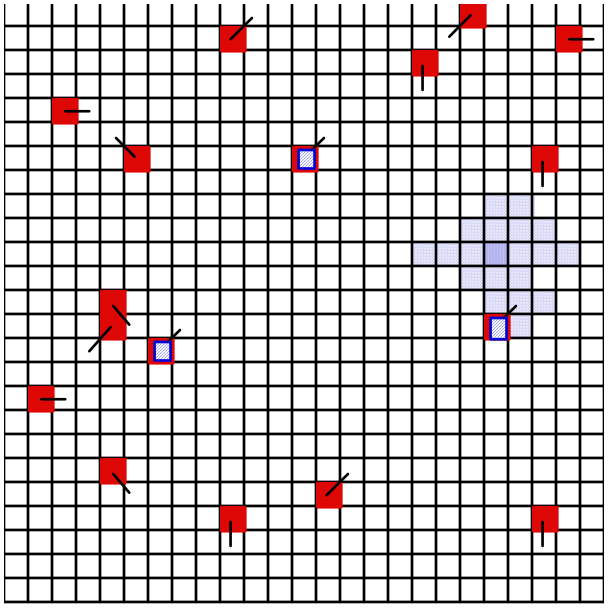
# EMERGENCE OF SORTING BEHAVIOR IN ANTS

## Epoch 1181 for best-of-generation individual from generation 27



# EMERGENCE OF SORTING BEHAVIOR IN ANTS

## Epoch 2,700 for best-of-generation individual from generation 27

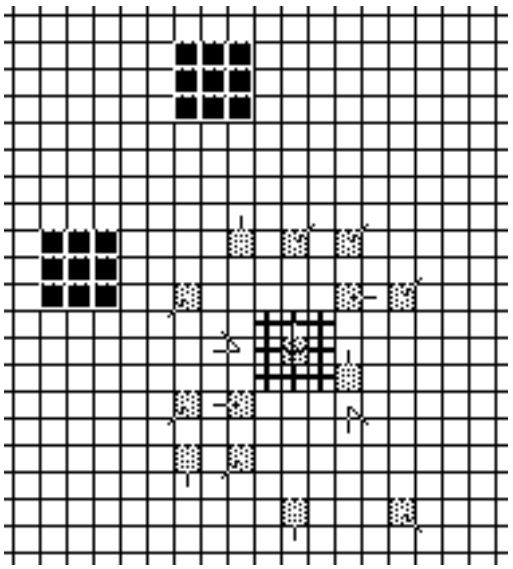


# TABLEAU FOR EMERGENT CENTRAL PLACE FOOD FORAGING BEHAVIOR

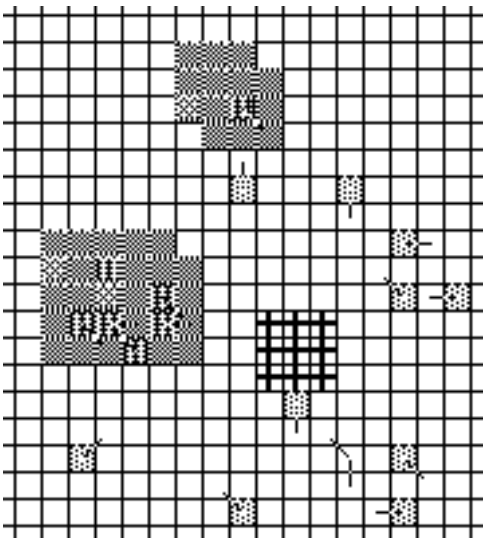
| | |
|---|---|
| **Objective:** | **Find a computer program which, when executed by a colony of 20 ants, causes emergent central place food foraging behavior in an ant colony.** |
| **Terminal set:** | `(MOVE-RANDOM), (MOVE-TO-NEST), (PICK-UP), (DROP-PHEROMONE).` |
| **Function set:** | `IF-FOOD-HERE, IF-CARRYING-FOOD,`<br>`MOVE-TO-ADJACENT-FOOD-ELSE,`<br>`MOVE-TO-ADJACENT-PHEROMONE-ELSE, PROGN.` |
| **Fitness cases:** | **One fitness case.** |
| **Raw fitness:** | **Number of food pellets (out of 144) transported to the nest within the allotted time.** |
| **Standardized fitness:** | **Total number of food pellets (144) minus raw fitness.** |
| **Hits:** | **Equals raw fitness for this problem.** |
| **Wrapper:** | **None.** |
| **Parameters:** | $M = 500$. $G = 51$. |

| **Success predicate:** | **An S-expression scores 144 hits.** |
|---|---|

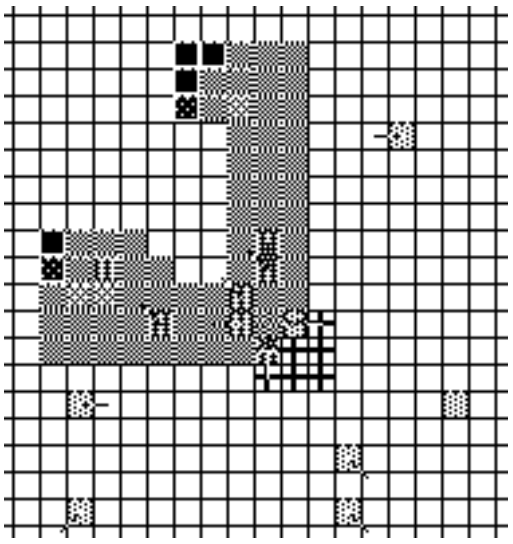
# CENTRAL PLACE FOOD FORAGING FIRST PHASE (STEP 3)



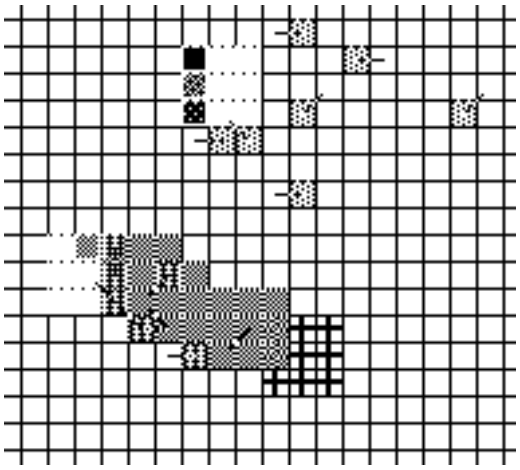
# CENTRAL PLACE FOOD FORAGING SECOND PHASE (STEP 12)



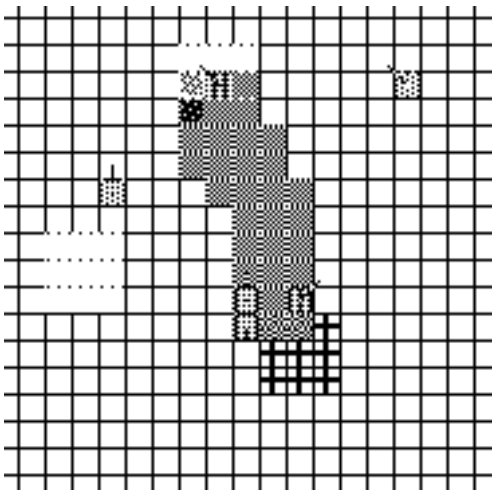
# CENTRAL PLACE FOOD FORAGING
# THIRD PHASE



# CENTRAL PLACE FOOD FORAGING PREMATURE DISINTEGRATION OF PHEROMONAL TRAIL (STEP 129)



# CENTRAL PLACE FOOD FORAGING
# STEP 152



# CENTRAL PLACE FOOD FORAGING

```
(PROGN (PICK-UP) (IF-CARRYING-FOOD (PROGN
(MOVE-TO-ADJACENT-PHEROMONE-ELSE (MOVE-
TO-ADJACENT-FOOD-ELSE (MOVE-TO-ADJACENT-
FOOD-ELSE (MOVE-TO-ADJACENT-FOOD-ELSE
(PICK-UP))))) (PROGN (PROGN (PROGN (PROGN
(MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP))
(PICK-UP)) (PROGN (MOVE-TO-NEST) (DROP-
PHEROMONE))) (PICK-UP)) (PROGN (MOVE-TO-
NEST) (DROP-PHEROMONE)))) (MOVE-TO-
ADJACENT-FOOD-ELSE (IF-CARRYING-FOOD
(PROGN (PROGN (DROP-PHEROMONE) (MOVE-TO-
ADJACENT-PHEROMONE-ELSE (IF-CARRYING-FOOD
(MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP))
(MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP)))))
(MOVE-TO-NEST)) (IF-FOOD-HERE (PICK-UP)
(IF-CARRYING-FOOD (PROGN (IF-FOOD-HERE
(MOVE-RANDOM) (IF-CARRYING-FOOD (MOVE-
RANDOM) (PICK-UP))) (DROP-PHEROMONE))
(MOVE-TO-ADJACENT-PHEROMONE-ELSE (MOVE-
RANDOM)))))))
```

# CENTRAL PLACE FOOD FORAGING

```
1 (PROGN (PICK-UP)
2   (IF-CARRYING-FOOD
3    (PROGN (MOVE-TO-ADJACENT-PHEROMONE-
ELSE
4       (MOVE-TO-ADJACENT-FOOD-ELSE
(PICK-UP)))
5        (MOVE-TO-ADJACENT-FOOD-ELSE
(PICK-UP))
6        (MOVE-TO-NEST)
7        (DROP-PHEROMONE)
8        (MOVE-TO-NEST)
9        (DROP-PHEROMONE))
10    (MOVE-TO-ADJACENT-FOOD-ELSE
11      (IF-FOOD-HERE
12       (PICK-UP)
13       (MOVE-TO-ADJACENT-PHEROMONE-ELSE
14        (MOVE-RANDOM)))))))
```

# TABLEAU FOR BLOCK STACKING

| | |
|---|---|
| **Objective:** | **Find a robotic plan for stacking blocks onto a STACK so that they spell UNIVERSAL.** |
| **Terminal set:** | **Sensors NN, TB, CS.** |
| **Function set:** | MS, MT, DU, NOT, EQ. |
| **Fitness cases:** | **A structured sample of 166 initial condition cases consisting of particular blocks on the STACK and TABLE.** |
| **Raw fitness:** | **The number of fitness cases for which the STACK equals UNIVERSAL after the S-expression is evaluated.** |
| **Standardized fitness:** | **Number of cases (i.e., 166) minus raw fitness.** |
| **Hits:** | **Equal to raw fitness for this problem.** |
| **Wrapper:** | **None.** |
| **Parameters:** | **$M = 500$. $G = 51$.** |
| **Success predicate:** | **An S-expression scores 166 hits.** |

# TABLEAU FOR THE RANDOMIZER PROBLEM

| | |
|---|---|
| **Objective:** | **Find an S-expression that takes the index J as its input and produces a high-entropy stream of random binary digits as its output.** |
| **Terminal set:** | **Index J and ℜ, the ephemeral random constant ℜ ranging over the small integers 0, 1, 2, and 3.** |
| **Function set:** | **+, -, *, QUOT%, MOD%.** |
| **Fitness cases:** | **16,384 ($2^{14}$) sequence positions.** |
| **Raw fitness:** | **Total entropy $E_{total}$.** |
| **Standardized fitness:** | **Maximum value of raw fitness (i.e., 28.000 bits) minus raw fitness.** |
| **Hits:** | **The integer just below 1,000 times raw fitness.** |
| **Wrapper:** | **Converts an positive value to a binary 1 and all other values to 0.** |

| **Parameters:** | ***M* = 500. *G* = 51.** |
|---|---|
| **Success predicate:** | **An S-expression scores 27,990 or more hits.** |

# TABLEAU FOR THE ONE-DIMENSIONAL CELLULAR AUTOMATON PROBLEM

| | |
|---|---|
| **Objective:** | **Find an S-expression that, when inserted at all cells in a one-dimensional cellular space, produces a high-entropy temporal stream of random binary digits as its output at a designated cell.** |
| **Terminal set:** | **X (center), W(west), and E (east) relative to each cell in question.** |
| **Function set:** | **AND, OR, NOT.** |
| **Fitness cases:** | **4,096 ($2^{12}$) temporal sequence steps.** |
| **Raw fitness:** | **Total entropy $E_{total}$ taken over temporal output for subsequences of length 4.** |
| **Standardized fitness:** | **Maximum value of raw fitness (i.e., 4.000 bits) minus raw fitness.** |
| **Hits:** | **1,000 times raw fitness.** |
| **Wrapper:** | **None.** |

| **Parameters:** | ***M* = 500. *G* = 51.** |
|---|---|
| **Success predicate:** | **An S-expression scores 3,990 or more hits.** |

# TABLEAU FOR TASK PRIORITIZATION (PAC MAN)

| | |
|---|---|
| **Objective:** | **Find a computer program for a Pac Man that scores the maximum number of points in the game.** |
| **Terminal set:** | `APILL`, `RPILL`, `DISPILL`, `AGA`, `RGA`, `DISGA`, `AGB`, `RGB`, `DISGB`, `AFOOD`, `DISU`, `AFRUIT`, `DISF`. |
| **Function set:** | `IFB`, `IFLTE`. |
| **Fitness cases:** | **One fitness case.** |
| **Raw fitness:** | **Points scored in the game.** |
| **Standardized fitness:** | **Standardized fitness is the maximum number of points (i.e., 18,220 ) minus raw fitness.** |
| **Hits:** | **Equals raw fitness for this problem.** |
| **Wrapper:** | **None.** |
| **Parameters:** | $M = 500$. $G = 51$. |

| | |
|---|---|
| **Success predicate:** | **None.** |

# TABLEAU FOR PROGRAMMATIC IMAGE COMPRESSION

| | |
|---|---|
| **Objective:** | **Find a LISP symbolic expression that returns the color value for each pixel in a two-dimensional image.** |
| **Terminal set:** | **$_{x,y}$, $\Re$, where the ephemeral random floating-point constant $\Re$ ranges over the interval [–1.0, +1.0].** |
| **Function set:** | **+, -, *, %.** |
| **Fitness cases:** | **Two-dimensional array of 900 pixels.** |
| **Raw fitness:** | **The sum, taken over the 900 fitness cases, of the absolute value of the difference between the color value produced by the S-expression for position ($_{x}$, $_{y}$) and the color value of the target image for position ($_{x,y}$).** |
| **Standardized fitness:** | **Same as raw fitness for this problem.** |

| | |
|---|---|
| **Hits:** | **Number of fitness cases for which the value of the wrapperized S-expression comes within 6 color values (out of 128) of the correct value.** |
| **Wrapper:** | **Converts arbitrary floating-point number into one of the 128 color values.** |
| **Parameters:** | ***M* = 2,000 (with over-selection). *G* = 51.** |
| **Success predicate:** | **An S-expression scores 900 hits.** |

# TARGET IMAGE FOR PROBLEM OF PROGRAMMATIC IMAGE COMPRESSION PRODUCED BY THE EXPRESSION $3 X^2 + 2 Y^2 - 0.85$ WRAPPER (OUTPUT INTERFACE) FOR PROGRAMMATIC IMAGE COMPRESSION

**Wrapper maps return value of program into [–1.0, +1.0] and thence into the desired range of 128 integral color values from 0 to 127:**

```
(* 64 (+ 1 (MAX -1.0 (MIN 1.0
S-EXPRESSION))))
```

# BEST-OF-GENERATION INDIVIDUAL FROM GENERATION 0 FOR PROGRAMMATIC IMAGE COMPRESSION

**Best-of-generation 0 – Fitness of 260.9 (over 900 fitness cases):**

```
(* (* (- (* (% Y X) X) (% (* Y
Y) (+ 0.0458984 -0.106705))) X)
X)
```

# BEST-OF-GENERATION INDIVIDUAL FROM GENERATION 6 FOR PROGRAMMATIC IMAGE COMPRESSION

**Best-of-generation 6 – Fitness of 18.93 (over 900 fitness cases):**

```
(+ (+ (+ (* (+ (* X X) (* Y Y))
(% Y Y)) (+ (* Y Y) -0.8116))
(+ 0.0458984-0.106705)) (* (% X
0.51979) X))
```

**Equivalent to the expression...**

```
(+ (* 2.9239 X X) (* 2 Y Y) -
0.8724)
```

# CUBE ROOT OF 2

- **Terminal set T = {ℜ}**
- **Function set F = {+, –, *, %},**

- **Typical S-expressions**

```
(* (* 0.537 -1.234) (+ 1.467
0.899))
```

- **Cubic equation $x^3 - 2 = 0$ has only one real root**

**$x = 2^{1/3} = 1.2599210498948732$**

- **Rewrite the ordinary cubic equation as the functional equation**

**$f_3(x) - 2 = 0$**

# CUBE ROOT OF 2

**Best-of-gen 0 equals 1.2473918 – 37 points:**

```
(- (% (% (* (% -0.119999945
0.9670001) 0.34300005) (% (* -
0.788 0.99100006) (% -
0.23699999 0.33200002)))
0.45500004) (* (% (- (- -
0.30699998 0.76300013) (+
0.5810001 0.85600007))
0.9390001) (- (* (* 0.6450001
0.82200015) 0.086000085) (- (*
0.549 0.9460001) 0.97300005))))
```

# CUBE ROOT OF 2

**Best-of-gen 2 equals 1.2602566 – 39 points:**

```
(+ (- 0.50600004 (+ -
0.045999944 (- (- -0.23699999
0.61100006) (% -0.059999943 -
0.26699996)))) (+ (* (- (-
0.8160001 -0.972) (% -0.83 -
0.811)) (- (* -0.09799993
0.42700005) (% -0.269 -0.822)))
(* (+ (* 0.411 -0.049999952)
0.4310001) (- (% -0.40199995
0.69500005) -0.37799996))))
```

# CUBE ROOT OF 2

## Best-of-gen 4 equals 1.2598813 – 89 points:

```
(- (+ (* (+ (% 0.15100002 (- (+ -
0.045999944 (- (- -0.23699999 0.61100006)
(% -0.059999943 -0.26699996))) (- (% (%
(* (% -0.119999945 0.9670001) 0.34300005)
(% (* -0.788 0.99100006) (% -0.23699999
0.33200002))) 0.45500004) (* (% (- (- -
0.30699998 0.76300013) (+ 0.5810001
0.85600007)) 0.9390001) (- (* (% (+
0.2570001 -0.706) (* 0.9130001 -0.847))
0.086000085) (- (* 0.549 0.9460001)
0.97300005))))))) 0.59800005) (+ (-
0.80200005 0.60800004) (+ 0.36800003 -
0.559))) -0.44799995) (+ (+ (- (* -0.861
0.9920001) 0.80700004) (* -0.09799993
0.42700005)) (* (* (* 0.10500002 0.314)
(- -0.74399996 0.12400007)) (+ -
0.69299996 (- 0.99600005 0.18700004)))))
```

# CUBE ROOT OF 2

## Best-of-gen 6 equals 1.2599242 – 111 points:

```
(- (+ (* (+ (% 0.15100002 (- (+ -0.045999944 (- (- -
0.23699999 0.61100006) (% -0.059999943 -0.26699996)))
(- (% (% (* (- (* (% (+ 0.2570001 -0.706) (*
0.9130001 -0.847)) 0.086000085) (- (* 0.549
0.9460001) 0.97300005)) 0.34300005) (% (+ (* (% -
0.119999945 0.9670001) (+ (- 0.80200005 0.60800004)
(+ 0.36800003 -0.559))) -0.44799995) (% -0.23699999
0.33200002))) 0.45500004) (* (% (- (- -0.30699998
0.76300013) (+ 0.5810001 0.85600007)) 0.9390001) (-
(* (% (+ 0.2570001 -0.706) (* 0.9130001 -0.847))
0.086000085) (- (* 0.549 0.9460001) 0.97300005))))))
0.59800005) (+ (- 0.80200005 0.60800004) (+
0.36800003 -0.559))) -0.44799995) (+ (+ (- (* -0.861
0.9920001) 0.80700004) (* -0.09799993 0.42700005)) (*
(* (* 0.10500002 0.314) (- -0.74399996 0.12400007))
(+ -0.69299996 (- 0.99600005 0.18700004)))))
```

# CUBE ROOT OF 2

## Best-of-gen 14 equals 1.2599211 - 139 points:

```
(- (+ (* (+ (% 0.15100002 (- (+ -0.045999944 (- (* (*
(* 0.10500002 0.314) (- -0.74399996 0.12400007)) (+ -
0.69299996 (- 0.99600005 0.18700004))) -0.23699999))
(- (% (% (* (- (* (% (+ 0.2570001 -0.706) (*
0.9130001 -0.847)) 0.086000085) (- (* 0.549
0.9460001) 0.97300005)) 0.34300005) (% (+ (* (% -
0.119999945 0.9670001) (+ (- 0.80200005 0.60800004)
(+ 0.36800003 -0.559))) -0.44799995) (% -0.23699999
0.33200002))) 0.45500004) (* (% (- (- -0.30699998
0.76300013) (+ 0.5810001 0.85600007)) 0.9390001) (-
(+ (+ (- (* -0.861 0.9920001) 0.80700004) (* -
0.09799993 0.42700005)) (* (* (* 0.10500002 0.314) (-
-0.74399996 0.12400007)) (+ -0.69299996 (- 0.99600005
0.18700004)))) (- (+ (- (* -0.861 0.9920001)
0.80700004) (* -0.09799993 0.42700005))
0.97300005))))))) 0.59800005) (+ (- 0.80200005
0.60800004) (+ 0.36800003 -0.559))) -0.44799995) (+
(+ (- (* -0.861 0.9920001) 0.80700004) (* -0.09799993
0.42700005)) (* (* (* 0.10500002 0.314) (- -
0.74399996 0.12400007)) (+ -0.69299996 (- 0.99600005
0.18700004)))))
```

# CUBE ROOT OF 2

| Generation | Value of individual S-expression | Difference from root of 1.2599211 | Structural complexity of S-expression |
|---|---|---|---|
| **0** | **1.2473918** | **–0.0125293** | 37 |
| **2** | **1.2602566** | **+0.0003355** | 39 |
| **4** | **1.2598813** | **–0.00000398** | 89 |
| **6** | **1.2599242** | **+0.0000031** | 111 |
| **14** | **1.2599211** | 0.0000000 | 139 |

# CUBE ROOT OF 2

- **Cube root of 2 is 1.2599210498948732**
- **Program equals 1.2599210498949058**
- **Correct to 14 decimal places**
- **Fitness 0.00000000000003264**

```
(+ (* (* -0.5403 (+ 0.5741 -0.8861)) (% (* 0.29690000000000016
0.08089999999999997) (+ (% (% (- -0.5962000000000001 0.3902000000000001) (-
(+ (% (* (+ (* 0.23550000000000004 0.15060000000000007) (* (* -
0.10289999999999999 -0.7332) 0.7723)) (* 0.23550000000000004
0.15060000000000007)) (+ 0.6026 (+ (+ (% (- 0.37250000000000005 -
0.34909999999999997) (- -0.776 -0.6013)) (- -0.5250999999999999 -
0.009000000000000008)) (% (- 0.29690000000000016 -0.34909999999999997) (- -
0.776 -0.6013))))) (* (+ -0.8861 (% -0.06019999999999992
0.051100000000000145)) (% -0.06019999999999992 0.051100000000000145))) (% -
0.49659999999999993 0.4475))) (+ (% (% (* (+ -0.1943999999999999
0.4366000000000001) (* 0.23550000000000004 0.15060000000000007)) (+ 0.6026
(* (* (+ (* -0.5403 -0.017199999999999993) (% -0.06019999999999992
0.051100000000000145)) (% (* (+ -0.1943999999999999 0.4366000000000001) (*
0.23550000000000004 0.15060000000000007)) (% (% 0.42100000000000004 -
0.4275) (- -0.48160000000000003 0.5708)))) 0.7723))) (- -0.8395 -0.1986))
(% (- 0.37250000000000005 -0.34909999999999997) (- -0.776 -0.6013)))) (% (%
(+ 0.6698000000000002 0.8714000000000002) (% (- -0.829 -0.636) (-
0.7635000000000001 -0.15899999999999992))) (- (- (* -0.5403 -
0.017199999999999993) (- -0.8395 -0.1986)) (- (* (* -0.5403 -
0.017199999999999993) (- 0.6004 -0.4343)) (- -0.951 (* (% 0.7803 0.9777)
0.31920000000000015)))))))) (+ (* (* -0.5403 -0.017199999999999993) -
0.19240000000000002) (+ (+ -0.13339999999999996 0.7944) 0.6004)))
```

# TABLEAU FOR EMPIRICAL DISCOVERY OF ECONOMETRIC EXCHANGE EQUATION

| | |
|---|---|
| **Objective:** | **Find an econometric model for the price level, in symbolic form, that fits a given sample of 80 actual quarterly data points.** |
| **Terminal set:** | **GNP82, FM2, FYGM3, ℜ, where the ephemeral random floating-point constant ℜ ranges over the interval [–1.000, +1.000].** |
| **Function set:** | **+, -, *, %, EXP, RLOG.** |
| **Fitness cases:** | **The given sample of 80 quarterly data points.** |
| **Raw fitness:** | **The sum, taken over 80 quarters, of the squares of differences between the S-expression for the price level expressed in terms of the three independent variables and the actual GD time series.** |

| | |
|---|---|
| **Standardized fitness:** | **Equals raw fitness for this problem.** |
| **Hits:** | **Number of fitness cases for which the S-expression comes within 1% of the actual value of the GD time series.** |
| **Wrapper:** | **None.** |
| **Parameters:** | **$M = 500$. $G = 51$.** |
| **Success predicate:** | **An S-expression scores 80 hits.** |

# ECONOMETRIC EXCHANGE EQUATION

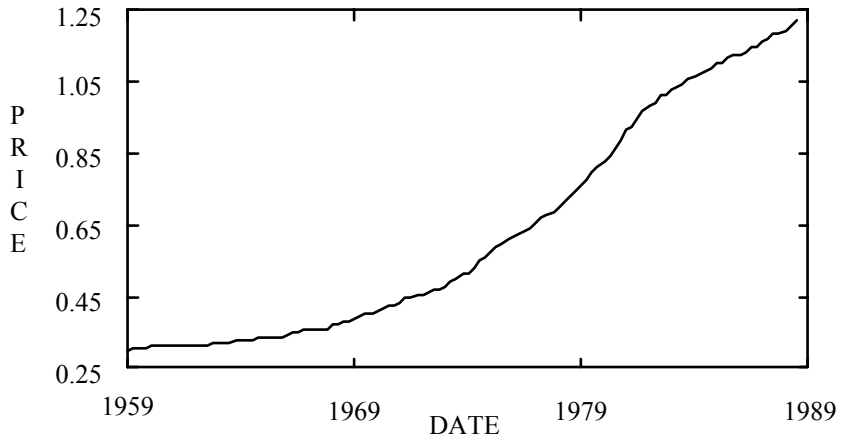
**120 actual quarterly values (from 1959:1 to 1988:4) of:**

- **the annual rate for the United States Gross National Product in billions of 1982 dollars (called GNP82)**
- **the Gross National Product Deflator (normalized to 1.0 for 1982) (called GD),**
- **the monthly values of the seasonally adjusted money stock M2 in billions of dollars, averaged for each quarter (called M2)**
- **the monthly interest rate yields of 3-month Treasury bills, averaged for each quarter (called FYGM3)**

# ECONOMETRIC EXCHANGE EQUATION

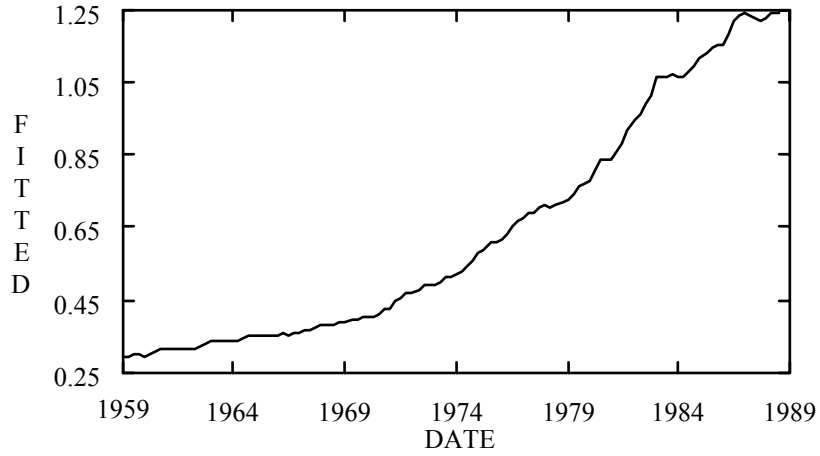
$$GD = \frac{(1.6527 * M2)}{GNP82}$$

# GROSS NATIONAL PRODUCT DEFLATOR (GD) FROM 1959:1 TO 1988:4



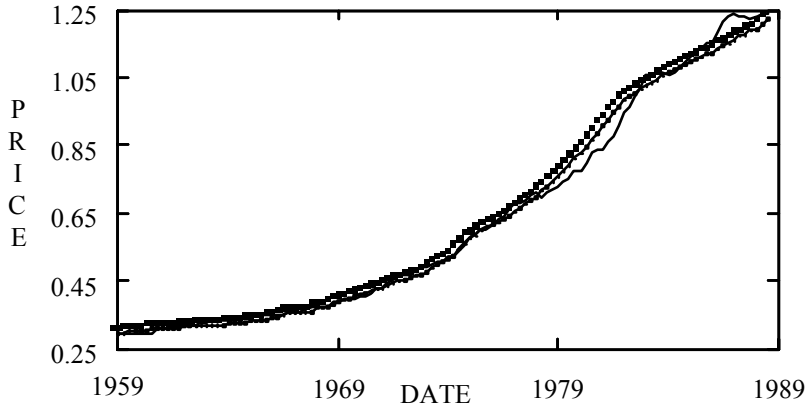
# ECONOMETRIC EXCHANGE EQUATION

# FITTED GD TIME SERIES



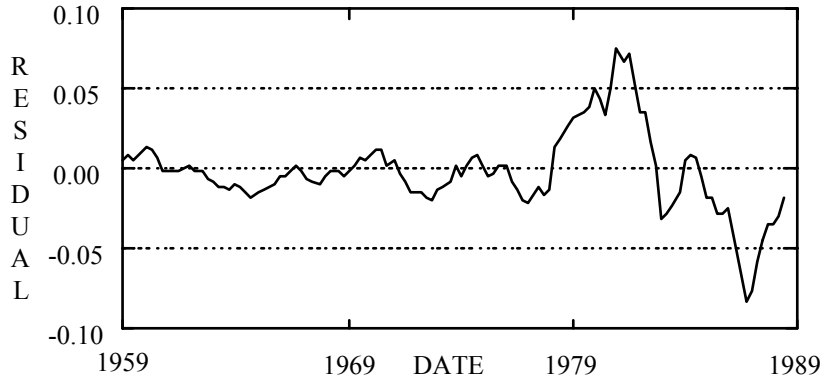
# ECONOMETRIC EXCHANGE EQUATION

# GROSS NATIONAL PRODUCT DEFLATOR GD OVERLAID WITH FITTED TIME SERIES



# ECONOMETRIC EXCHANGE EQUATION

# RESIDUALS BETWEEN GROSS NATIONAL PRODUCT DEFLATOR GD AND THE FITTED TIME SERIES



# ECONOMETRIC EXCHANGE EQUATION

## Best-of-run individual

```
(% (+ (* (+ (* -0.402 -0.583)
(% FM2 (- GNP82 (- 0.126 (+ (+
-0.83 0.832) (% (% GNP82 (* (-
0.005 GNP82) (% GNP82
GNP82)))0.47)))))) FM2) FM2)
GNP82)
```

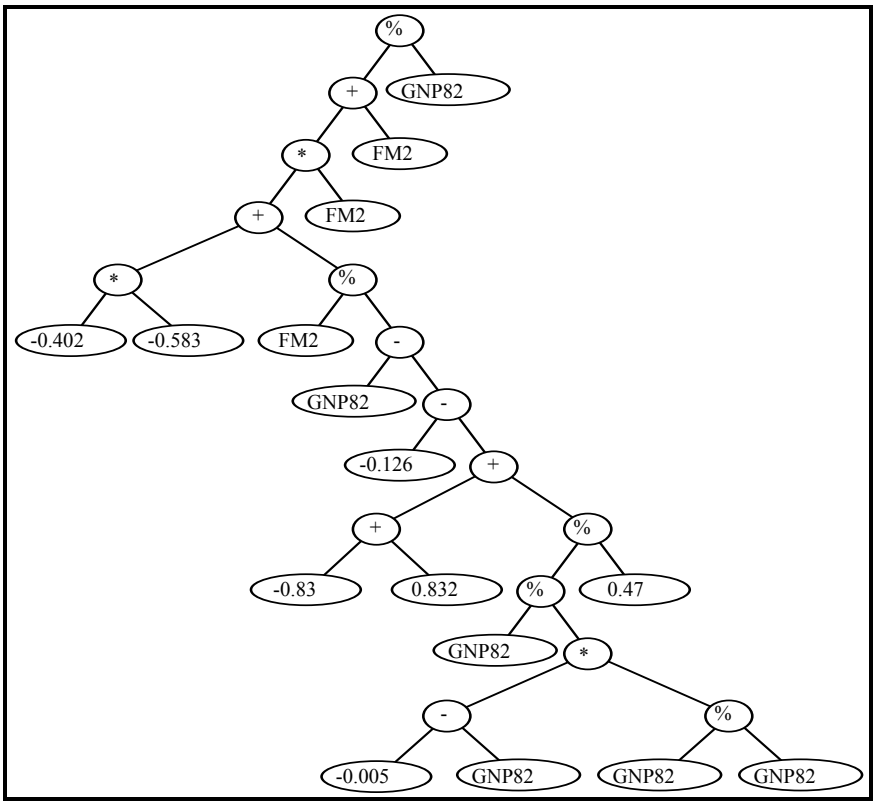
## Sum of squared errors of 0.009272

## Equivalent to...

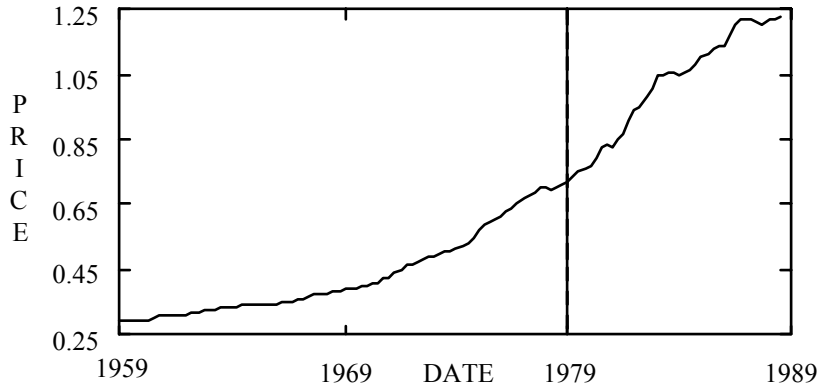
$$GD = \frac{(1.634 * M2)}{GNP82}$$

# ECONOMETRIC EXCHANGE EQUATION

# BEST-OF-RUN INDIVIDUAL FOR THE PRICE LEVEL USING THE FIRST TWO-THIRDS OF DATA



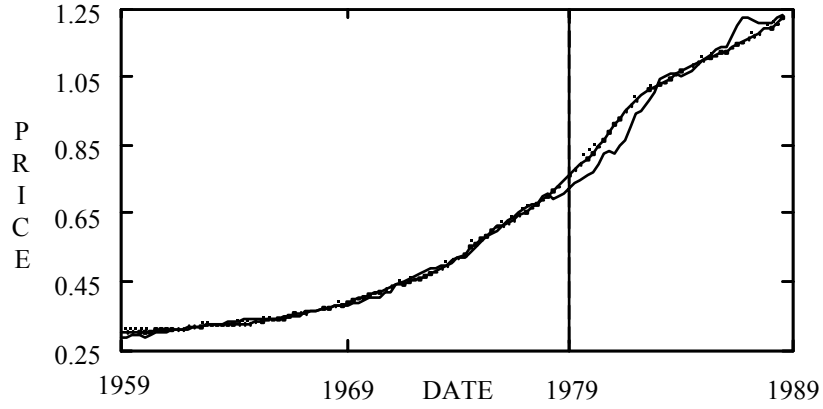
# GRAPH OF BEST-OF-RUN INDIVIDUAL FOR THE PRICE LEVEL USING THE FIRST TWO-THIRDS OF THE DATA



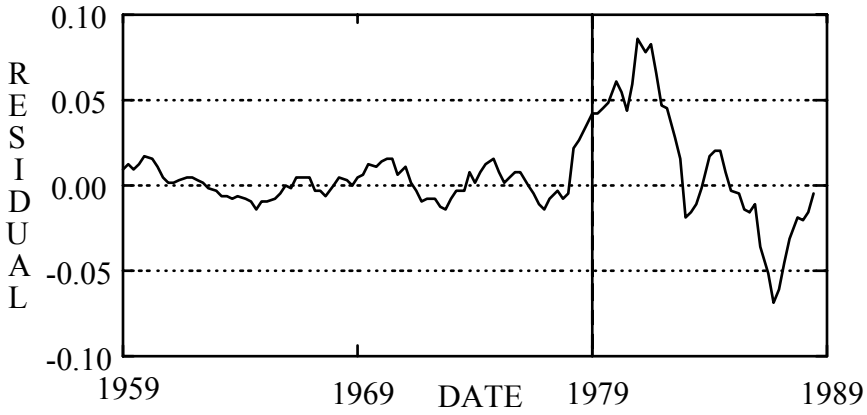

# SQUARED ERRORS AND CORRELATIONS USING THE FIRST TWO-THIRDS OF THE DATA

| Data range | 1- 120 | 1 - 80 | 81 - 120 |
|---|---|---|---|
| $R^2$ | 0.993480 | 0.997949 | 0.990614 |
| Sum of squared errors | 0.075388 | 0.009272 | 0.066116 |

# GD WITH FITTED TIME SERIES USING THE FIRST TWO-THIRDS OF THE DATA




# RESIDUALS



# GRAPH OF BEST-OF-RUN INDIVIDUAL FOR THE PRICE LEVEL USING THE LAST TWO-THIRDS OF THE DATA

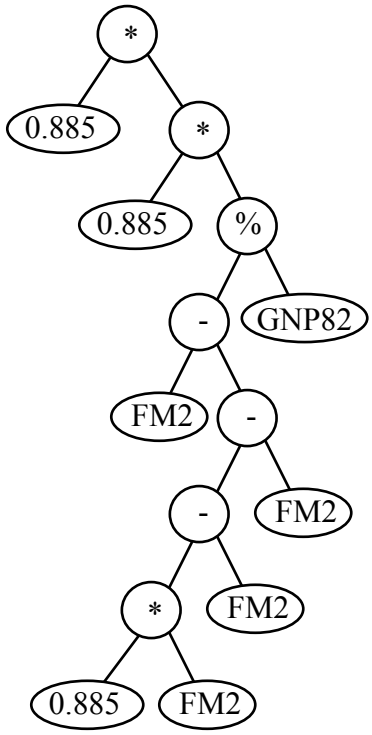
## Best-of-run individual:

```
(* 0.885 (* 0.885 (% (- FM2 (-
(- (* 0.885 FM2) FM2) FM2))
GNP82)))
```

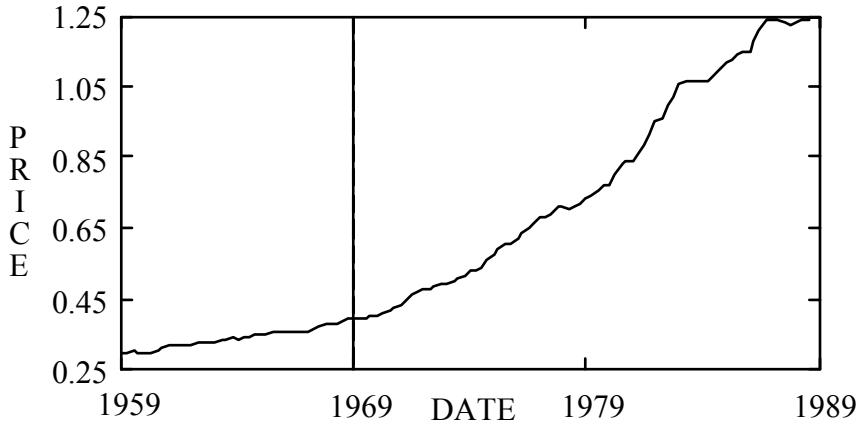
## Equivalent to ...

$$GD = \frac{(1.6565 * M2)}{GNP82}$$

# BEST-OF-RUN INDIVIDUAL FOR THE PRICE LEVEL USING THE LAST TWO-THIRDS OF THE DATA



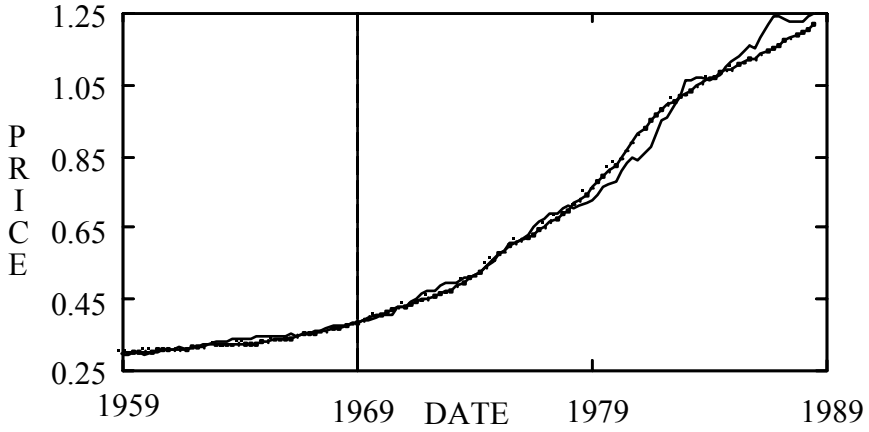
# GRAPH OF BEST-OF-RUN INDIVIDUAL FOR THE PRICE LEVEL USING THE LAST TWO-THIRDS OF THE DATA



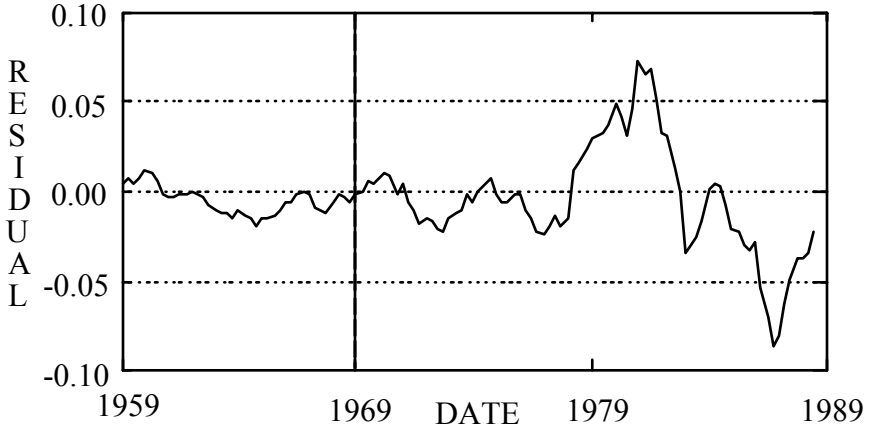

# SQUARED ERRORS AND CORRELATIONS USING THE LAST TWO-THIRDS OF THE DATA

| Data range | 1- 120 | 1 - 40 | 41 - 120 |
|---|---|---|---|
| $R^2$ | 0.993130 | 0.999136 | 0.990262 |
| Sum of squared errors | 0.079473 | 0.003225 | 0.076247 |

# GD WITH FITTED TIME SERIES USING THE LAST TWO-THIRDS OF THE DATA




# RESIDUALS



# FOOD FORAGING STRATEGY OF THE CARIBBEAN *ANOLIS* LIZARD

**Courtesy of Jonathan Roughgarden, *Theory of Population Genetics and Evolutionary Ecology: An Introduction* (1979).**

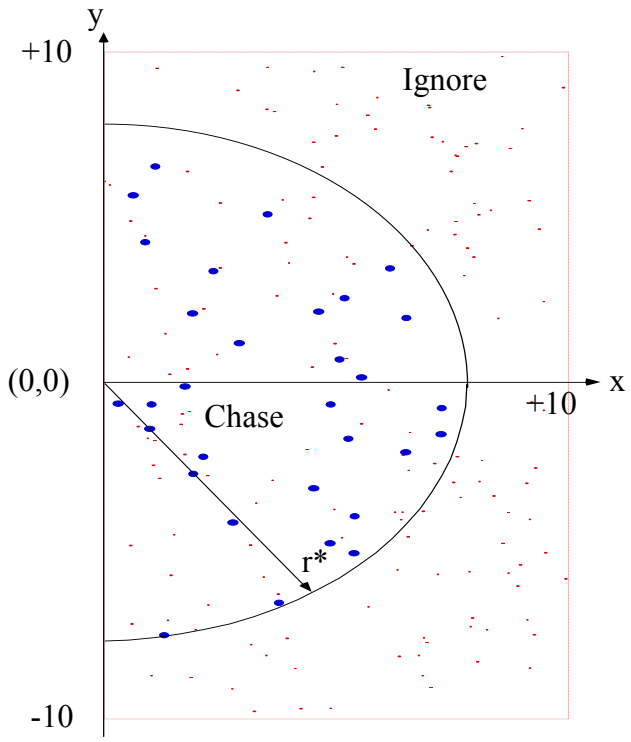
# FOOD FORAGING STRATEGY OF THE CARIBBEAN *ANOLIS* LIZARD

- **4-Dimensional Control Problem**
- **2 slowly-varying variables**
  - **sprint velocity, $v$**
  - **abundance of insects, $a$**
- **2 rapidly-varying variables**
  - **horizontal position, $x$**
  - **vertical position, $y$**
- **One binary control variable (Chase or Ignore)**

# SWITCHING CURVE AND OPTIMAL FORAGING STRATEGY

**Sig [cube root($\frac{3v}{\pi a}$) – square root($x^2 + y^2$)]**

**$r^*$ = cube root($\frac{3v}{\pi a}$)**

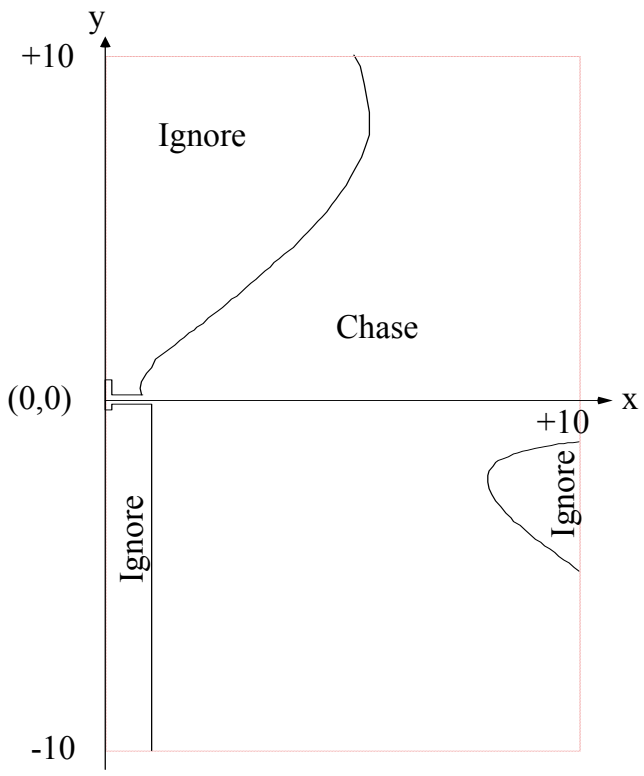


# TABLEAU FOR FOOD FORAGING STRATEGY OF *ANOLIS* LIZARD

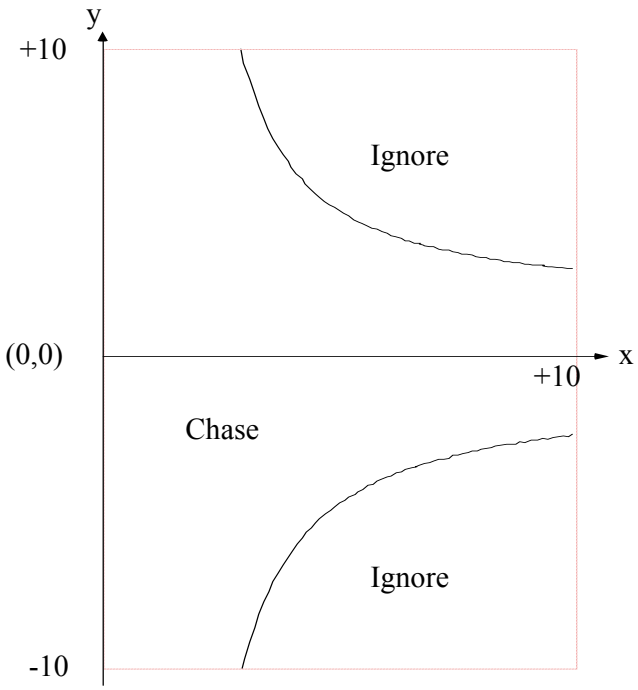
| | |
|---|---|
| **Objective:** | **Find a control strategy enabling a lizard to maximize food by deciding whether to chase or ignore insects alighting within its territory.** |
| **Terminal set:** | **X, Y, AB, VEL, and the ephemeral random constant ℜ ranging from –1.000 to +1.000.** |
| **Function set:** | **+, -, *, %, SREXPT, IFLTE.** |
| **Fitness cases:** | **Two 300-second experiments for each of 36 combinations of value of abundance AB and sprint velocity VEL.** |
| **Raw fitness:** | **The sum, taken over the 72 fitness cases, of the number of insects eaten by the lizard when the lizard chases or ignores insects in accordance with the S-expression.** |

| | |
|---|---|
| **Standardized fitness:** | **The maximum value of raw fitness (17,256) minus the raw fitness of the S-expression.** |
| **Hits:** | **Number of fitness cases for which the number of insects eaten is equal to or greater than one less than the number eaten using the closed-form optimal foraging strategy.** |
| **Wrapper:** | **Converts any non-negative value returned by an S-expression to +1 and converts all other values to –1.** |
| **Parameters:** | ***M* = 1,000 (with tournament selection). *G* = 61.** |
| **Success predicate:** | **An S-expression scores 72 hits.** |

# SWITCHING CURVES OF A PROGRAM FROM THE 65TH PERCENTILE OF FITNESS FOR GENERATION 0 FOR VERSION 1 FOR $A$ = 0.003 AND = 1.5



# SWITCHING CURVE OF THE BEST-OF-GENERATION PROGRAM FROM GENERATION 10 FOR VERSION 1 FOR $A = 0.003$ AND $V = 1.5$



# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 1

## • GENERATION 60 – 67 points – 1,652 insects – 60 hits (with hits criterion of 1)

```
(+ (- (+ (- (SREXPT AB -0.9738)
(* (SREXPT X X) (* X AB))) (*
(+ VEL AB) (% (- VEL (% AB Y))
(+ 0.7457 0.338898)))) (SREXPT
Y X)) (- (- (SREXPT AB -0.9738)
(SREXPT -0.443604 (- (- (+ (-
(SREXPT AB -0.9738) (SREXPT -
0.443604 Y)) (+ AB X)) (SREXPT
Y X)) (* (* (+ X 0.0101929) AB)
X)))) (* (SREXPT Y Y) (* X
AB))))
```

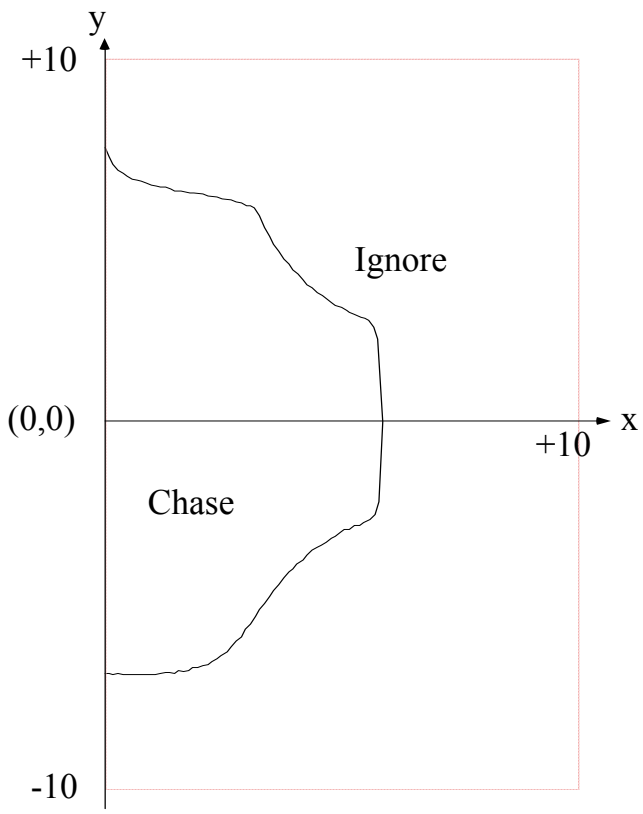
## Equivalent to ...

$$-0.44^{a} + x + a^{-0.9738} - (0.44y + y^{x} + ax[x + 0.01])$$

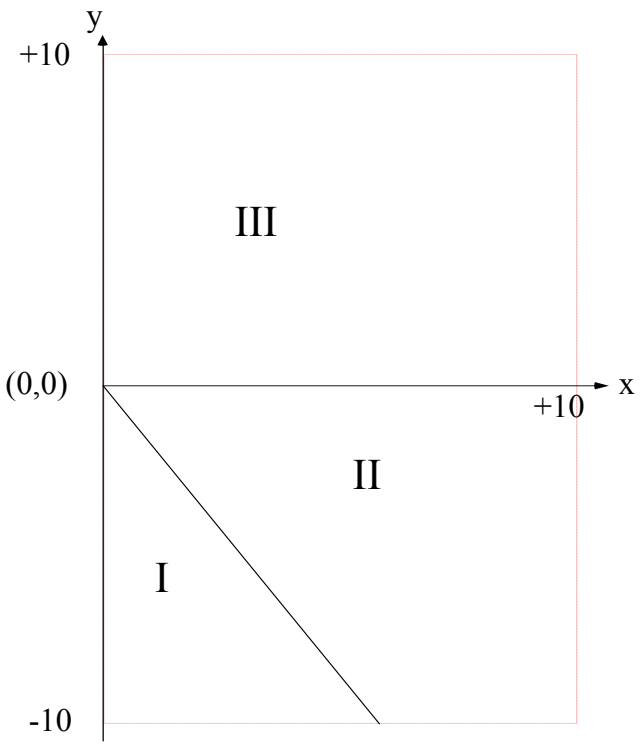
$$+ 0.922(v + a)\left(v - \frac{a}{y}\right)$$

$$+ 2a^{-0.97} - y^{x} - ax(x^{x} + yy)$$

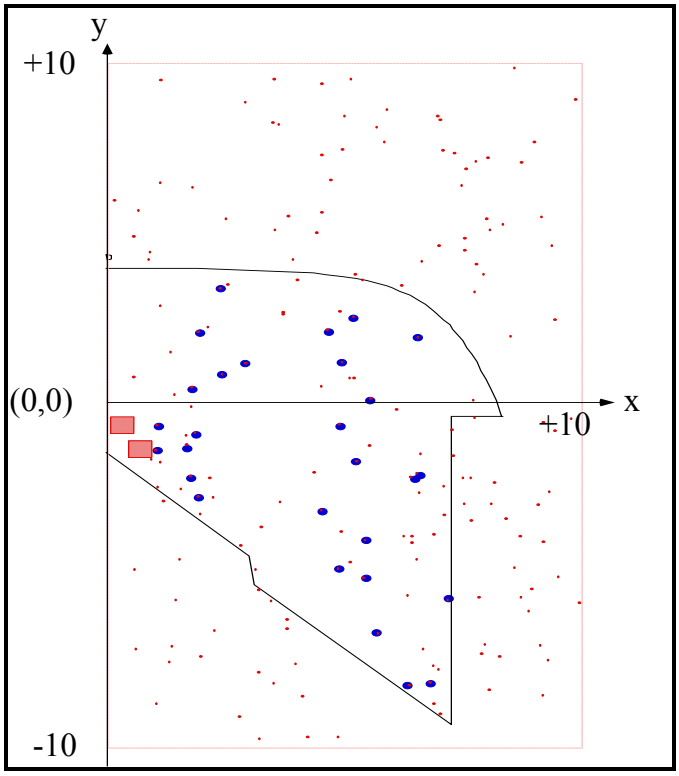
# SWITCHING CURVE OF THE BEST-OF-GENERATION PROGRAM FROM GENERATION 60 FOR VERSION 1 FOR *A* = 0.003 AND = 1.5



# THREE REGIONS FOR VERSION 2 OF THE FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 2



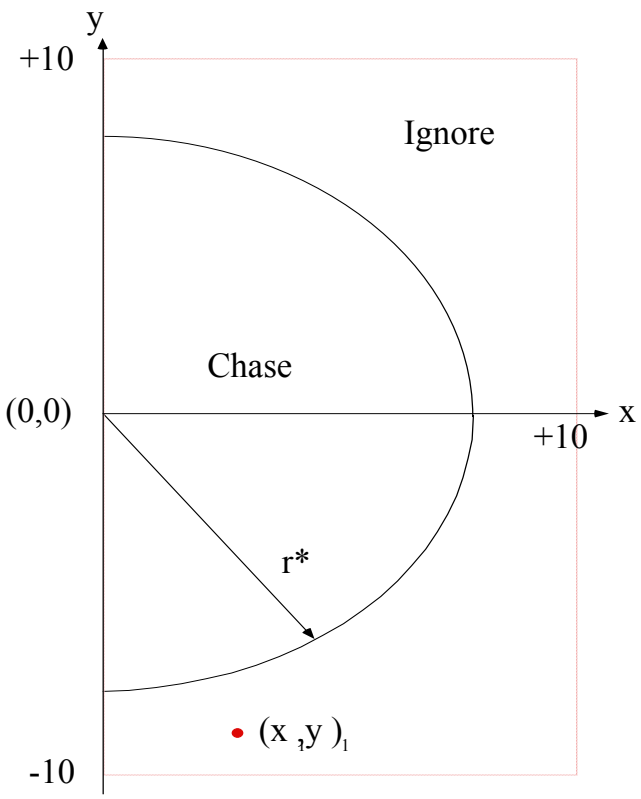
# SWITCHING CURVE OF THE BEST-OF-GENERATION PROGRAM FROM GENERATION 46 FOR VERSION 2 FOR *A* = 0.003 AND = 1.5

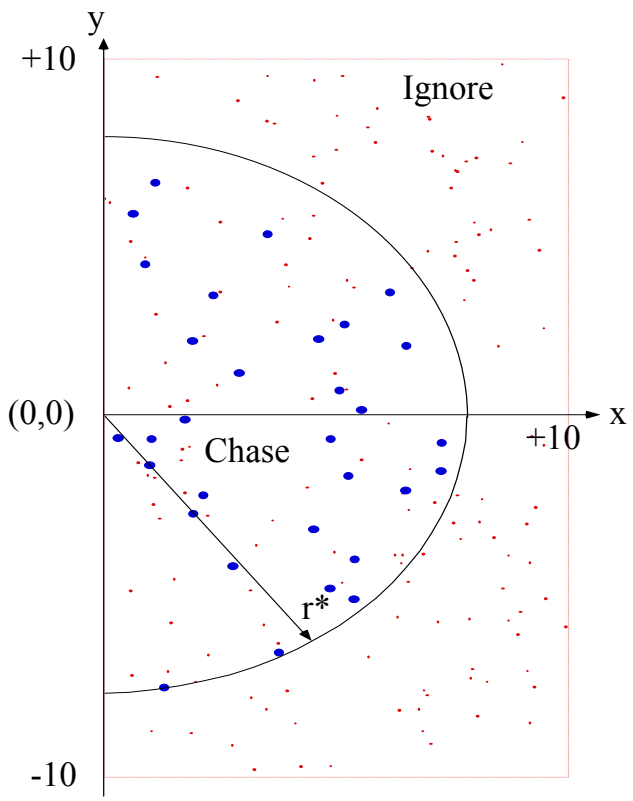


# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD

$$\text{Sig}\left[\text{cube root}\left(\frac{3v}{\pi a}\right) - \text{square root}(x^2 + y^2)\right]$$

$$r^* = \text{cube root}\left(\frac{3v}{\pi a}\right)$$


y
+10
Ignore
Chase
(0,0)
x
+10
r*
(x ,y )1
-10


y
+10
Ignore
(0,0)
x
+10
Chase
r*
-10

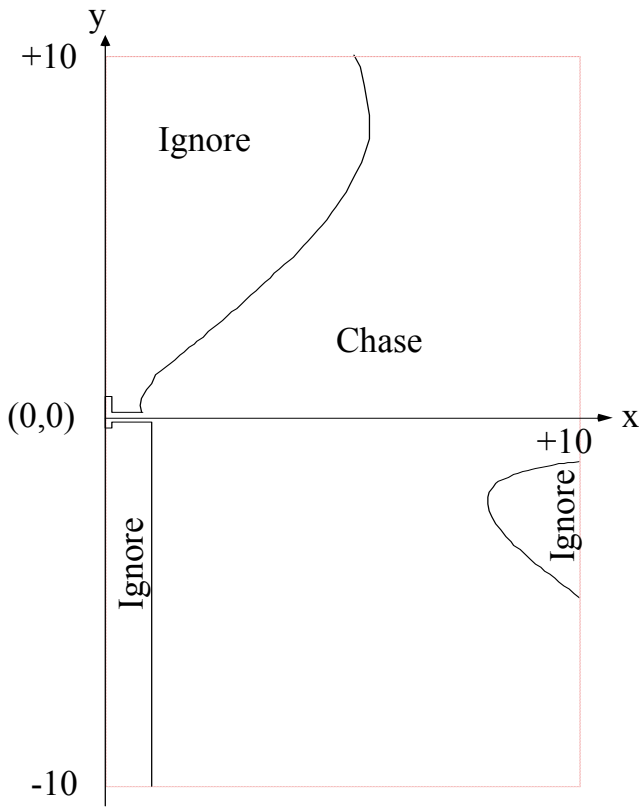
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 1

**GENERATION 0 –– 65th PERCENTILE OF FITNESS**

```
(+ (- (- (* (SREXPT VEL Y) (+ -0.3752s0 X)) (+ (* VEL 0.991s0)
(+ -0.9522s0 X))) (IFLTE (+ (S% AB Y) (S% VEL X)) (+ (+ X
0.3201s0) (S% AB VEL)) (IFLTE (IFLTE X AB X Y) (SREXPT AB VEL)
(+ X -0.9962s0) (S% -0.0542984s0 AB)) (- (* Y Y) (* Y VEL))))
(S% (IFLTE (IFLTE (+ X Y) (+ X Y) (+ VEL AB) (* Y Y)) (- (S%
0.662094s0 AB) (* VEL X)) (+ (SREXPT AB X) (- X Y)) (IFLTE (*
Y Y) (SREXPT VEL VEL) (+ Y VEL) (IFLTE AB AB X VEL))) (IFLTE
(IFLTE (SREXPT X AB) (* VEL -0.0304031s0) (IFLTE 0.9642s0 X Y
AB) (SREXPT 0.0341034s0 AB)) (+ (- VEL 0.032898s0) (- X VEL))
(IFLTE (- X Y) (SREXPT VEL 0.141296s0) (* X AB) (SREXPT -
0.6911s0 0.5399s0)) (SREXPT (+ AB AB) (IFLTE 0.90849s0 VEL AB
0.9308s0)))))
```

**143 points**
**Caught 1,235 insects**

# SWITCHING CURVES – 65TH PERCENTILE– GENERATION 0 – VERSION 1 FOR $A = 0.003$ AND $= 1.5$



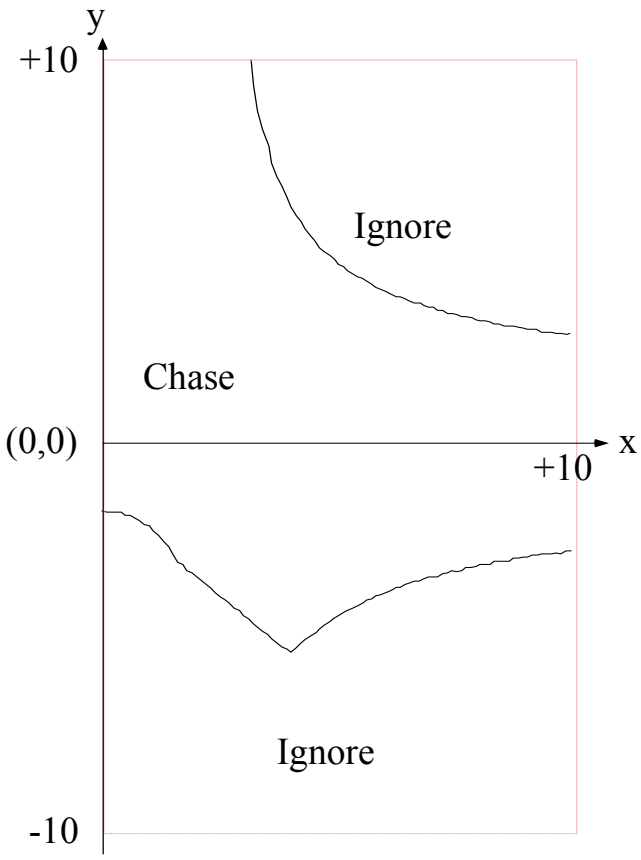
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 1

**GENERATION 0 –– BEST-OF-GENERATION INDIVIDUAL**

```
(- (- (+ (* 0.5605 Y) (% VEL VEL)) (* (SREXPT Y X) (* X AB)))
(* (* (+ X 0.0101929) (* -0.155502 X)) (IFLTE (+ VEL Y) (- AB
X) (* X Y) (SREXPT VEL X))))
```

**37 points**
**Caught 1,460 insects**

# SWITCHING CURVES – BEST OF GENERATION 0 – VERSION 1 – $A$ = 0.003 AND = 1.5



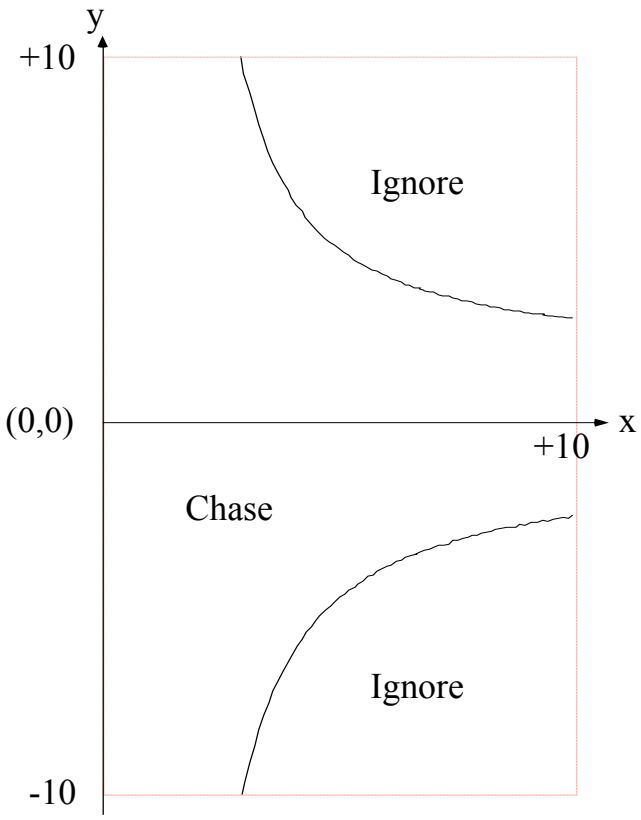
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 1

**GENERATION 10**

```
(- (- X (* (SREXPT Y X) (* X AB))) (* (* (+ X 0.0101929) (* -
0.155502 (+ AB X))) (IFLTE (+ X (+ (- (SREXPT X Y) (+ X
0.240997)) (+ 0.105392 VEL))) (% VEL 0.8255) (* (SREXPT X VEL)
(+ -0.7414 VEL)) (SREXPT VEL X))))
```

**47 points**
**Caught 1,514 insects**

# SWITCHING CURVES –BEST OF GENERATION 10 – VERSION 1 FOR *A* = 0.003 – = 1.5



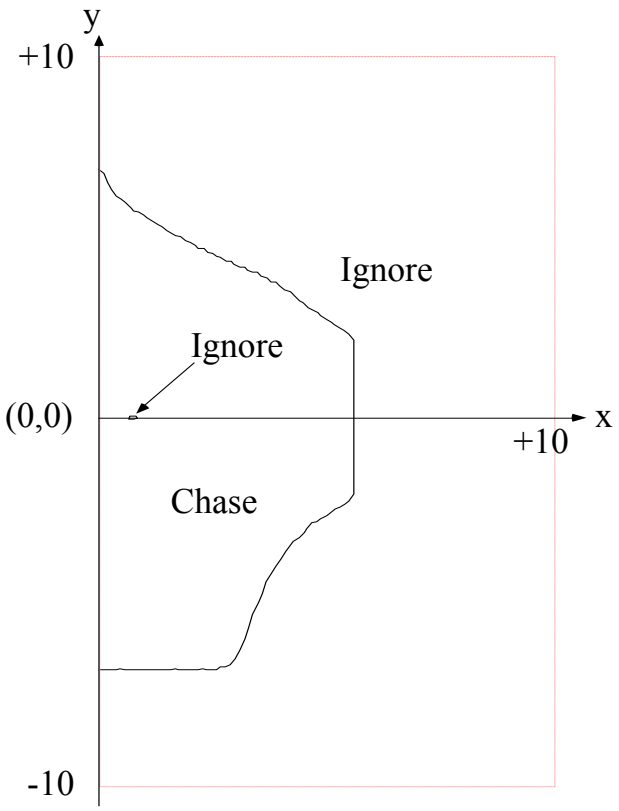
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 1

**GENERATION 25**

```
(- (- (+ (- (- (- (SREXPT AB -0.9738) (SREXPT -0.443604 Y)) (*
(SREXPT Y (+ (* (SREXPT (% (SREXPT Y AB) (- VEL -0.9724)) (+ X
0.0101929)) 0.457596) (+ Y X))) (* X AB))) (* (* (+ X
0.0101929) (% (+ Y -0.059105) (* 0.9099 Y))) (IFLTE (+ X
(SREXPT AB Y)) (% VEL 0.8255) (IFLTE Y VEL 0.282303 -0.272697)
(SREXPT (* (SREXPT Y X) (* X AB)) X)))) (% AB 0.412598)) (*
(SREXPT X X) (* X AB))) 0.4662)
```

**81 points**
**Caught l,514 insects**
**52 hits**

# SWITCHING CURVES –BEST OF GENERATION 25 – VERSION 1 FOR *A* = 0.003 – = 1.5



# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 1

**GENERATION 60**

```
(+ (- (+ (- (SREXPT AB -0.9738) (*
(SREXPT X X) (* X AB))) (* (+ VEL AB) (%
(- VEL (% AB Y)) (+ 0.7457 0.338898))))
(SREXPT Y X)) (- (- (SREXPT AB -0.9738)
(SREXPT -0.443604 (- (- (+ (- (SREXPT AB
-0.9738) (SREXPT -0.443604 Y)) (+ AB X))
(SREXPT Y X)) (* (* (+ X 0.0101929) AB)
X)))) (* (SREXPT Y Y) (* X AB))))
```

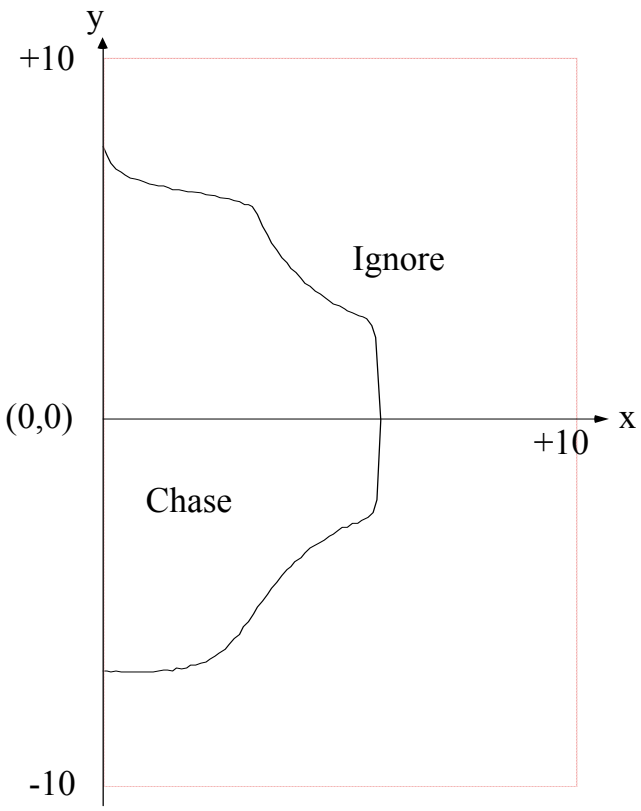
**67 points**
**Caught 1,652 insects – 60 hits (with hits criterion of 1)**

$$-0.44a + x + a^{-0.9738} - (0.44y + y^x + ax[x + 0.01])$$

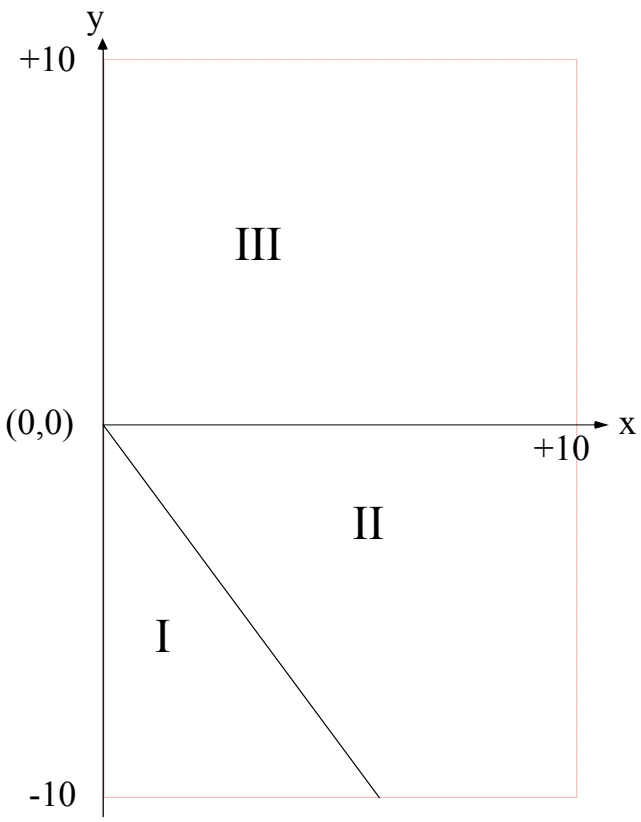
$$+ 0.922(v + a)(v - \frac{a}{y})$$

$$+ 2a^{-0.97} - y^x - ax(x^x + yy)$$

# SWITCHING CURVES –BEST OF GENERATION 60 – VERSION 1 FOR *A* = 0.003 – = 1.5



# *ANOLIS* LIZARD – VERSION 2



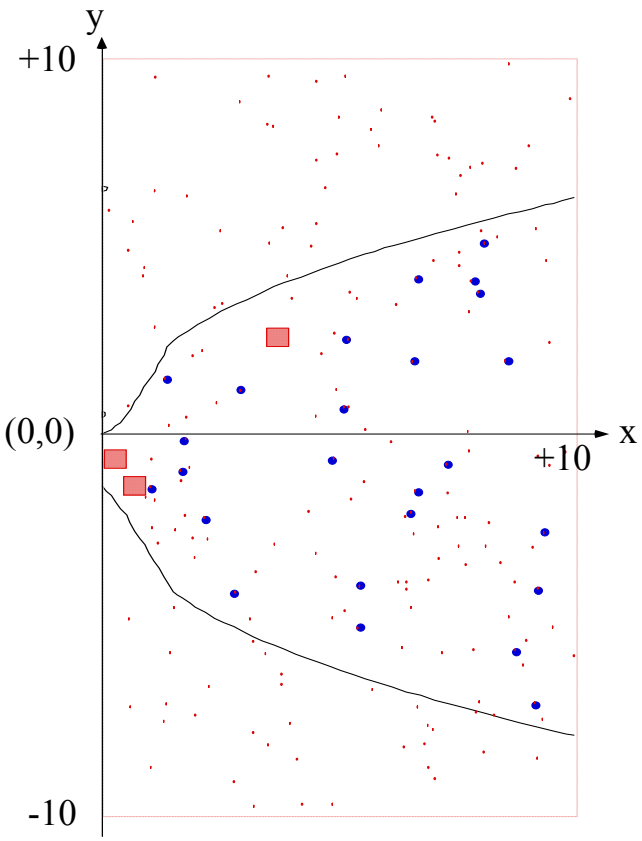
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 2

**GENERATION 0 – BEST-OF-GENERATION INDIVIDUAL**

```
(+ (S% (* (IFLTE X VEL VEL X)
(+ VEL Y)) (- (S% AB X) (+ Y
Y))) (SREXPT (* (SREXPT VEL
VEL) (S% X Y)) (+ (IFLTE AB VEL
0.194s0 X) (IFLTE VEL Y VEL
VEL))))
```

**37 points**
**1,164 insects**

# SWITCHING CURVES –BEST OF GENERATION 0 – VERSION 2 – $A$ = 0.003 – = 1.5



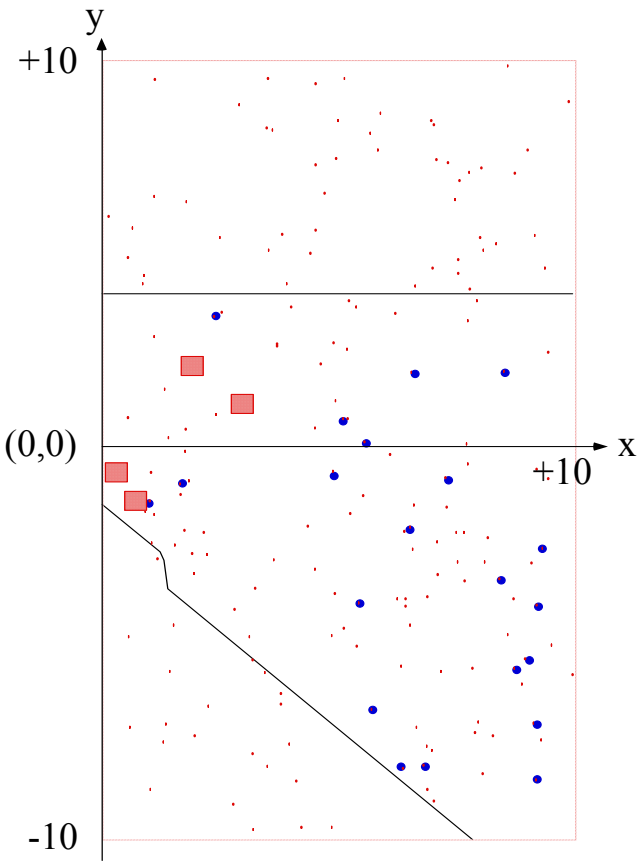
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 2

**GENERATION 12**

```
(IFLTE AB (* (SREXPT (* X 0.71089s0)
(IFLTE VEL AB 0.053299s0 X)) (+ (- VEL Y)
(IFLTE (* (IFLTE (SREXPT (S% -0.175102s0
(SREXPT Y AB)) (SREXPT (S% VEL AB) (IFLTE
Y (+ 0.175598s0 Y) (+ VEL (+ X (+ AB Y)))
(* 0.7769s0 (IFLTE 0.7204s0 AB 0.962204s0
AB))))) VEL (- VEL (SREXPT (S% VEL AB)
(S% 0.8029s0 0.36119s0))) (+ -0.157204s0
X)) VEL) (- X X) (+ VEL 0.8965s0) (*
0.180893s0 AB)))) (+ (IFLTE (- VEL X) (S%
-0.588s0 Y) (SREXPT 0.5443s0 -0.6836s0)
(S% X X)) (S% (+ X Y) (- VEL AB))) (- (S%
(SREXPT AB Y) (IFLTE Y Y X Y)) (IFLTE VEL
AB X X)))
```

107 POINTS – CAUGHT 1,228 INSECTS

# SWITCHING CURVES –BEST OF GENERATION 12 – VERSION 2 – $A$ = 0.003 – = 1.5



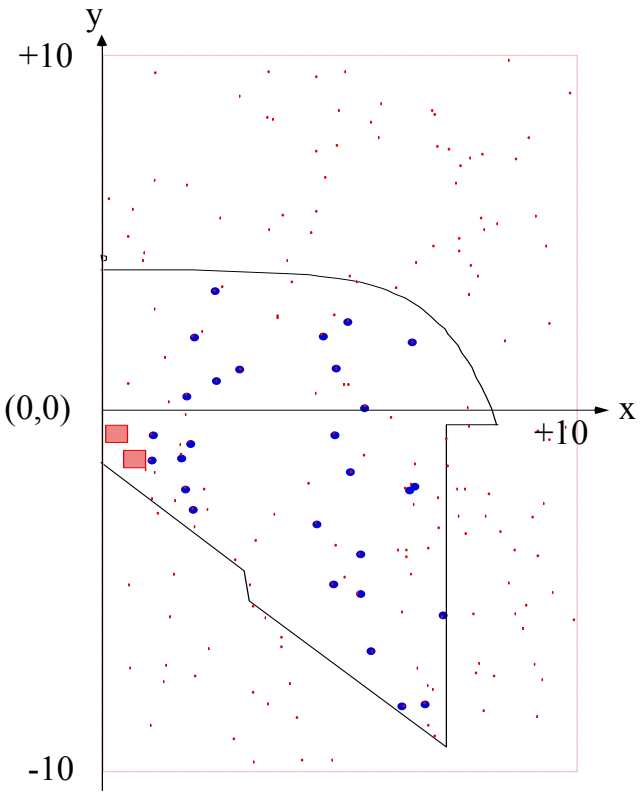
# FOOD FORAGING STRATEGY FOR THE CARIBBEAN *ANOLIS* LIZARD – VERSION 2

**GENERATION 46**

```
(IFLTE AB (* (SREXPT (* -0.588s0 0.71089s0) (IFLTE
VEL AB 0.053299s0 X)) (+ (- VEL Y) (IFLTE (S% (SREXPT
AB Y) (IFLTE Y Y X Y)) (- VEL (IFLTE X (+ (* AB AB)
(+ (IFLTE AB X AB AB) (* VEL AB))) (* (- X X) AB)
AB)) (+ VEL 0.8965s0) (+ 0.175598s0 Y)))) (+ (IFLTE
(+ VEL VEL) X (SREXPT 0.5443s0 -0.6836s0) (S% X X))
(S% (+ X Y) (- VEL AB))) (- (* (IFLTE (SREXPT (SREXPT
-0.0914s0 Y) (IFLTE (+ X Y) (S% -0.588s0 Y) (* AB
0.304092s0) (S% X X))) X (- VEL (IFLTE Y AB X Y)) (+
-0.157204s0 (IFLTE (+ (- (S% (S% X (- VEL (IFLTE Y AB
X Y))) (- (S% (SREXPT AB Y) (IFLTE Y Y X Y)) (IFLTE
VEL AB X X))) (+ (SREXPT Y Y) (- -0.172798s0 Y)))
(IFLTE (SREXPT AB X) (- (S% AB 0.7782s0) 0.444794s0)
(* (IFLTE Y (SREXPT (SREXPT X 0.299393s0) (+ VEL X))
0.6398s0 Y) (- X -0.6541s0)) (IFLTE (- (SREXPT Y
0.4991s0) (- (S% (SREXPT AB Y) (IFLTE Y Y X Y))
(IFLTE VEL AB X X))) AB (SREXPT VEL (SREXPT (SREXPT X
0.299393s0) (* -0.3575s0 X))) (+ (S% VEL AB) X))))
VEL (- VEL (- (S% (SREXPT AB Y) (+ (SREXPT Y Y) (S%
VEL VEL))) (IFLTE VEL AB X X))) (+ -0.157204s0 X))))
VEL) (IFLTE VEL AB X X)))
```

227 POINTS – CAUGHT 1,379 INSECTS

# SWITCHING CURVES –BEST OF GENERATION 46 – VERSION 2 – $A$ = 0.003 – = 1.5

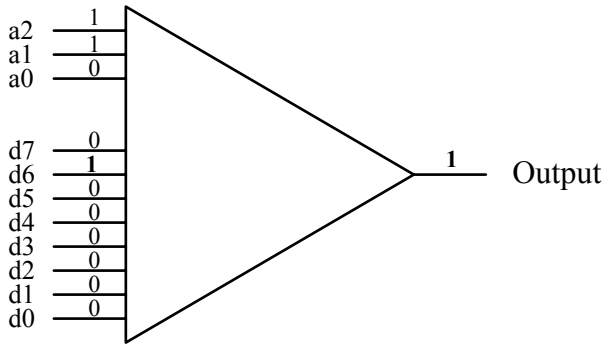


# TABLEAU FOR THE BOOLEAN 11-MULTIPLEXER PROBLEM

| | |
|---|---|
| **Objective:** | **Find a Boolean S-expression whose output is the same as the Boolean 11-multiplexer function.** |
| **Terminal set:** | `A0, A1, A2, D0, D1, D2, D3, D4, D5, D6, D7.` |
| **Function set:** | `AND, OR, NOT, IF.` |
| **Fitness cases:** | **The $2^{11} = 2{,}048$ combinations of the 11 Boolean arguments.** |
| **Raw fitness:** | **Number of fitness cases for which the S-expression matches correct output.** |
| **Standardized fitness:** | **Sum, taken over the $2^{11} = 2{,}048$ fitness cases, of the Hamming distances (i.e., number of mismatches). Standardized fitness equals 2,048 minus raw fitness for this problem.** |
| **Hits:** | **Equivalent to raw fitness for this problem.** |
| **Wrapper:** | **None.** |

| | |
|---|---|
| **Parameters:** | ***M*** **= 4,000 (with over-selection).** ***G*** **= 51.** |
| **Success Predicate:** | **An S-expression scores 2,048 hits.** |

# BOOLEAN 11-MULTIPLEXER



# BOOLEAN 11-MULTIPLEXER - GENERATION 0

- **Contradictory / Constant**

```
(AND A0 (NOT A0))
```

- **Inefficient**

```
(OR D7 D7)
```

- **Wrong arguments**

```
(IF D0 A0 A2)
```

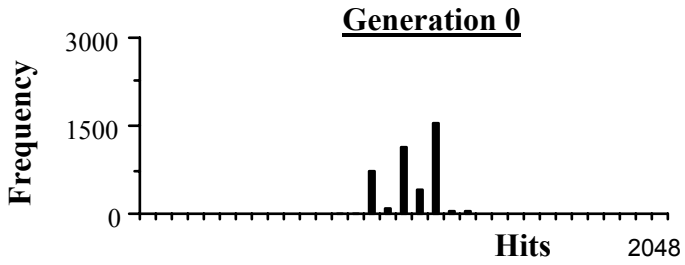
- **Nonsense**

```
(IF (IF (IF D2 D2 D2) D2 D2) D2 D2)
```

- **Worst of Generation 0 – 768 mismatches (i.e., 1,280 matches or hits out of 2,048)**

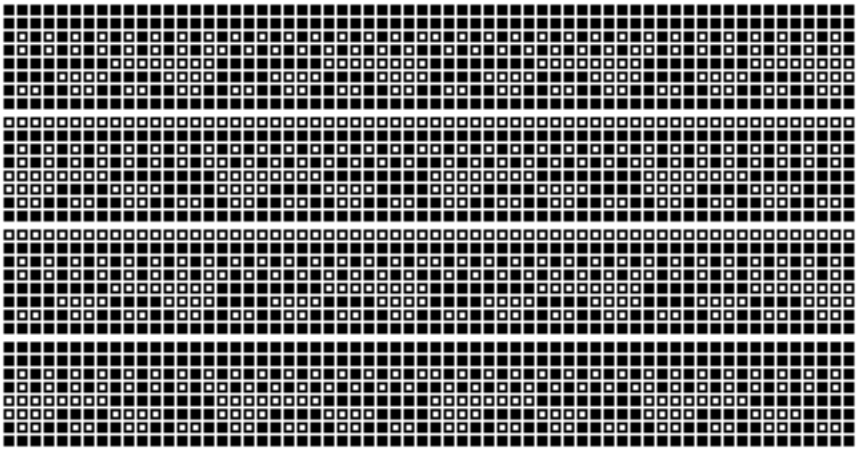
```
(OR (NOT A1) (NOT (IF (AND A2 A0) D7 D3)))
```

# BOOLEAN 11-MULTIPLEXER - GENERATION 0

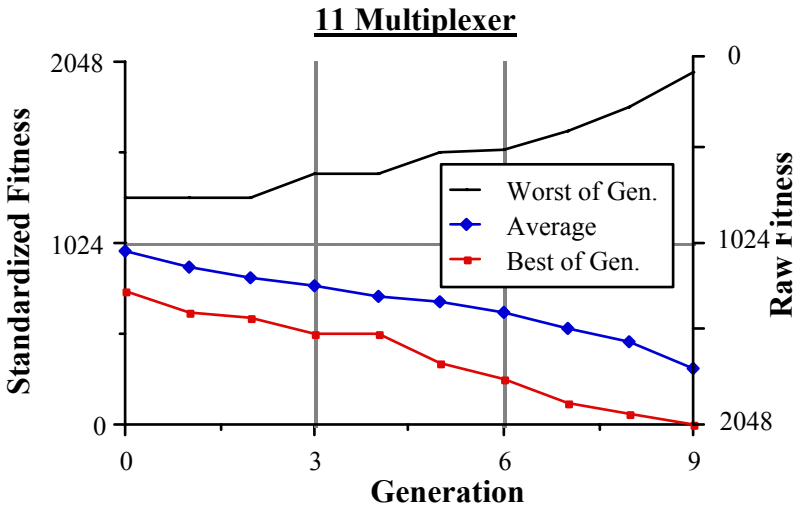


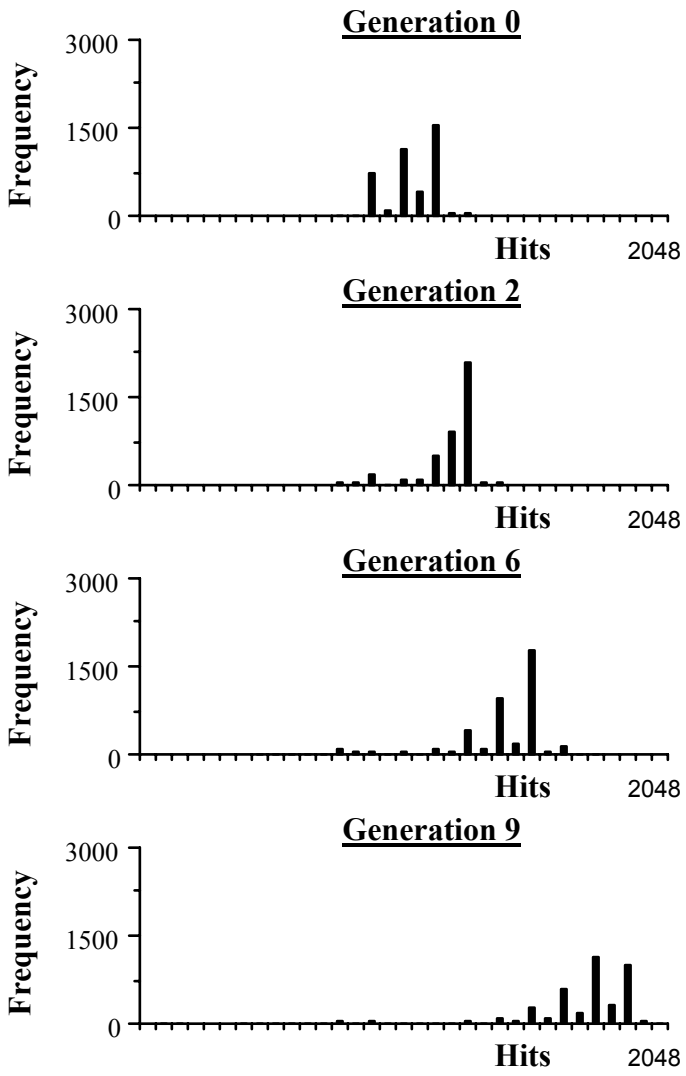
# BOOLEAN 11-MULTIPLEXER - GENERATION 0

```
(IF A0 D1 D2)
```



# BOOLEAN 11-MULTIPLEXER - FITNESS CURVES




Generation 0
Frequency
3000
1500
0
Hits
2048
Generation 2
Frequency
3000
1500
0
Hits
2048
Generation 6
Frequency
3000
1500
0
Hits
2048
Generation 9
Frequency
3000
1500
0
Hits
2048

# BOOLEAN 11-MULTIPLEXER

- **Best of Generation 1 - SF = 640**

```
(IF A0 (IF A2 D7 D3) D0)
```

- **Best of Generation 3 - SF = 576**

```
(IF A2 (IF A0 D7 D4)(AND (IF
(IF A2 (NOT D5) A0) D3 D2) D2))
```

- **Best of Generation 5 - SF = 384**

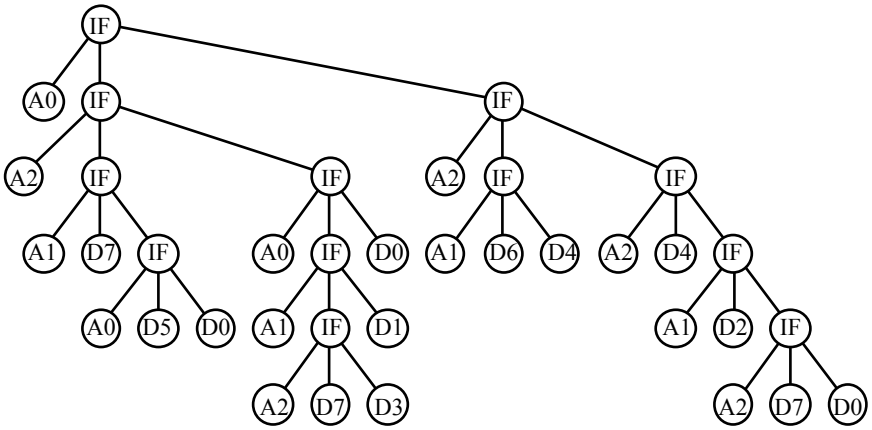
```
(IF A0 (IF A2 D7 D3) (IF A2 D4
(IF A1 D2 (IF A2 D7 D0))))
```

# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION

- **Best of Generation 9 - SF = 0 (2,048 hits)**

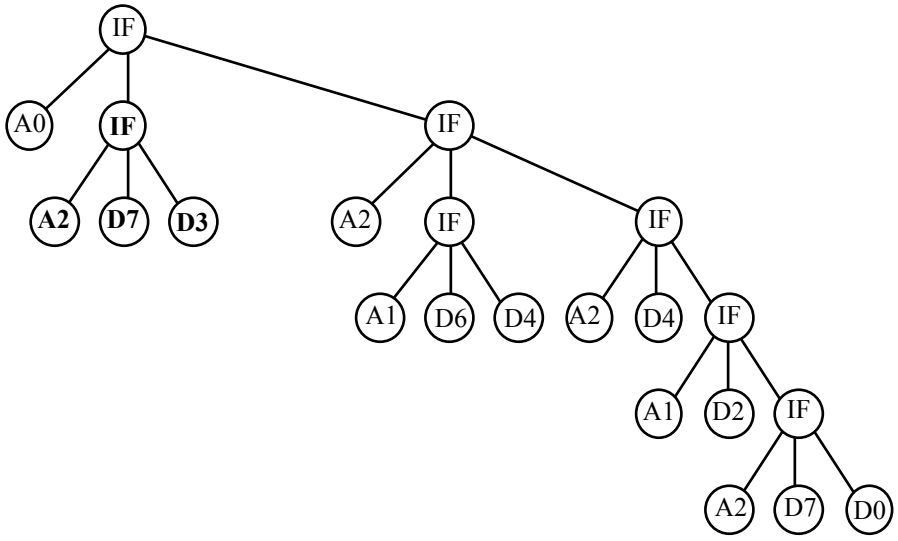
```
(IF A0 (IF A2 (IF A1 D7 (IF A0
D5 D0)) (IF A0 (IF A1 (IF A2 D7
D3) D1) D0)) (IF A2 (IF A1 D6
D4) (IF A2 D4 (IF A1 D2 (IF A2
D7 D0)))))
```

# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION



# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION

- **Parent "A" of Generation 8 - SF = 256 (1,792 hits) – 58th best**



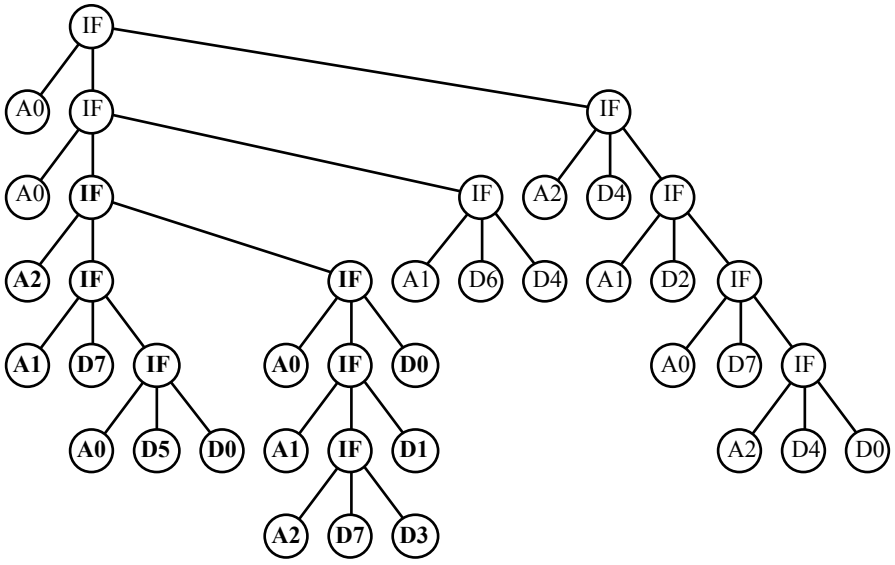
# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION

- **Parent "A" of Generation 8 - SF = 256 (1,792 hits) – 58th best**

```
(IF A0 (IF A2 D7 D3)
    (IF A2 (IF A1 D6 D4) (IF A2
D4 (IF A1 D2 (IF A2 D7 D0)))))
```

# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION

## • Parent "B" of Generation 8 - SF = 128 (1,920 hits) - Best of Generation 8

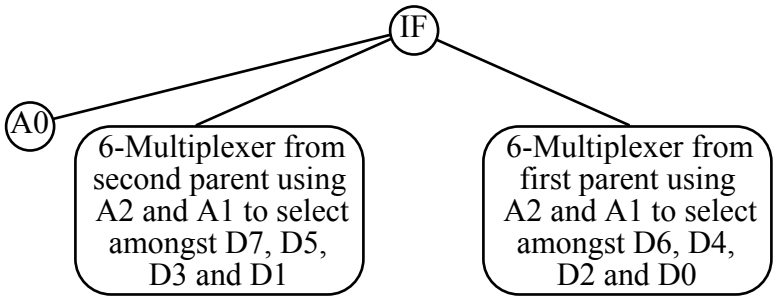


# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION

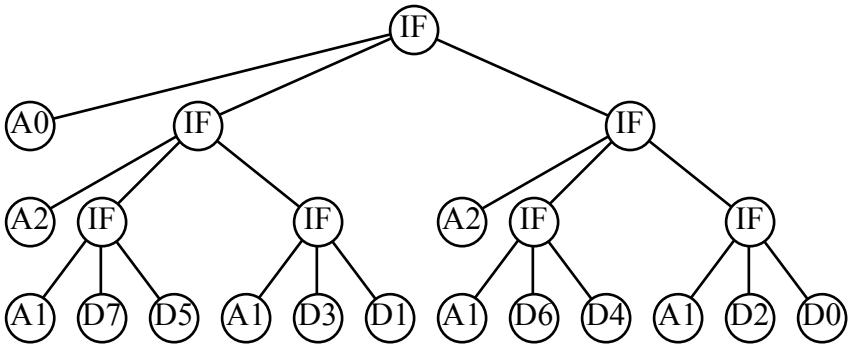
- **Parent "B" of Generation 8 - SF = 128 (1,920 hits) - Best of Generation 8**

```
(IF A0 (IF A0 (IF A2 (IF A1 D7
(IF A0 D5 D0)) (IF A0 (IF A1
(IF A2 D7 D3) D1) D0))
        (IF A1 D6 D4)) (IF A2 D4
(IF A1 D2 (IF A0 D7 (IF A2 D4
D0)))))
```

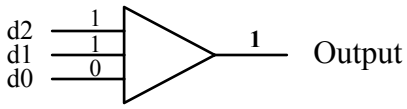
# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION



# BOOLEAN 11-MULTIPLEXER – 100% CORRECT SOLUTION



# BOOLEAN EVEN-3-PARITY FUNCTION




| Fitness case | D2 | D1 | D0 | Even-3-parity |
|---|---|---|---|---|
| 0 | NIL | NIL | NIL | T |
| 1 | NIL | NIL | T | NIL |
| 2 | NIL | T | NIL | NIL |
| 3 | NIL | T | T | T |
| 4 | T | NIL | NIL | NIL |
| 5 | T | NIL | T | T |
| 6 | T | T | NIL | T |
| 7 | T | T | T | NIL |

# GP TABLEAU WITHOUT ADFS FOR THE EVEN-3-PARITY PROBLEM

| | |
|---|---|
| **Objective:** | **Find a program that produces the value of the Boolean even-3-parity function as its output when given the value of the three independent Boolean variables as its input.** |
| **Terminal set without ADFs:** | **D0, D1, and D2.** |
| **Function set without ADFs:** | **AND, OR, NAND, and NOR.** |
| **Fitness cases:** | **All $2^3 = 8$ combinations of the three Boolean arguments D0, D1, and D2.** |
| **Raw fitness:** | **The number of fitness cases for which the value returned by the program equals the correct value of the even-3-parity function.** |

| | |
|---|---|
| **Standardized fitness:** | **The standardized fitness of a program is the sum, over the $2^3 = 8$ fitness cases, of the Hamming distance (error) between the value returned by the program and the correct value of the Boolean even-3-parity function.** |
| **Hits:** | **Same as raw fitness.** |
| **Wrapper:** | **None.** |
| **Parameters:** | **$M = 16{,}000$. $G = 51$.** |
| **Success predicate:** | **A program scores the maximum number of hits.** |

# BOOLEAN EVEN-3-PARITY

## GENERATION 5 - 45 POINTS - 8 HITS

```
(AND (OR (OR D0 (NOR D2 D1))
D2) (AND (NAND (NOR (NOR D0 D2)
(AND (AND D1 D1) D1)) (NAND (OR
(AND D0 D1) D2) D0)) (OR (NAND
(AND D0 D2) (OR (NOR D0 (OR D2
D0)) D1)) (NAND (NAND D1 (NAND
D0 D1)) D2))))
```

# BOOLEAN EVEN-4-PARITY

## GENERATION 5 - 149 POINTS - 16 HITS

```
(AND (OR (OR (OR (NOR D0 (NOR D2 D1))
(NAND (OR (NOR (AND D3 D0) D2) (NAND D0
(NOR D2 (AND D1 (OR D3 D2))))) D3)) (AND
(AND D1 D2) D0)) (NAND (NAND (NAND D3 (OR
(NOR D0 (NOR (OR D3 D2) D2)) (NAND (AND
(AND (AND D3 D2) D3) D2) D3))) (NAND (OR
(NAND (OR D0 (OR D0 D1)) (NAND D0 D1))
D3) (NAND D1 D3))) D3)) (OR (OR (NOR (NOR
(AND (OR (NOR D3 D0) (NOR (NOR D3 (NAND
(OR (NAND D2 D2) D2) D2)) (AND D3 D2)))
D1) (AND D3 D0)) (NOR D3 (OR D0 D2)))
(NOR D1 (AND (OR (NOR (AND D3 D3) D2)
(NAND D0 (NOR D2 (AND D1 D0)))) (OR (OR
D0 D3) (NOR D0 (NAND (OR (NAND D2 D2) D2)
D2)))))) (AND (AND D2 (NAND D1 (NAND (AND
D3 (NAND D1 D3)) (AND D1 D1)))) (OR D3
(OR D0 (OR D0 D1))))))
```

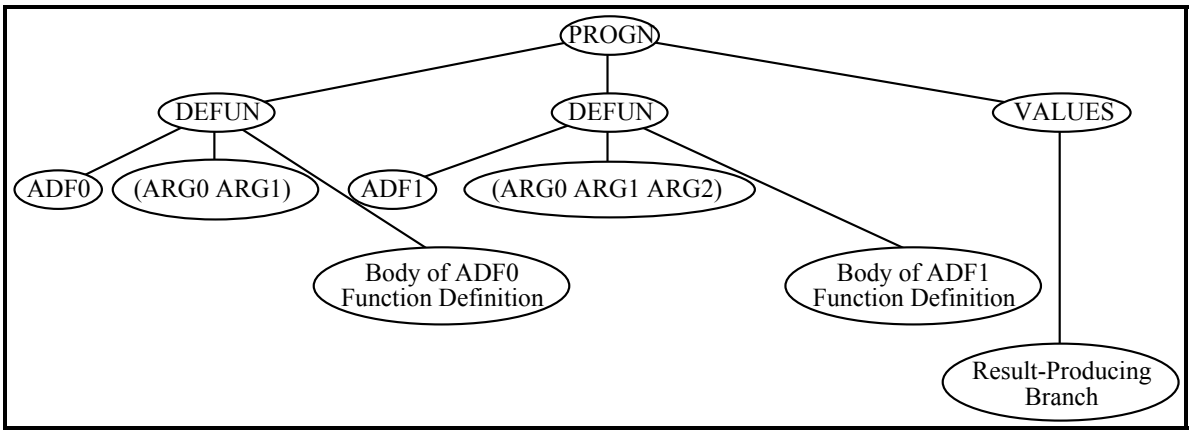
# BOOLEAN EVEN-5-PARITY

## GENERATION 27 - 347 POINTS - 32 HITS

```
(NAND (NAND (OR (AND (AND (AND (OR D2 D3) (OR D4 D2)) (AND (OR
D4 D0) (AND D1 D1))) (NOR (NOR D1 D4) (OR (NOR D3 D4) (NOR D0
D2)))) (NOR (NOR (NAND (AND (OR D1 D0) (OR D3 D2)) (OR D1 D3))
(NOR D1 D2)) (NOR (AND (NAND D4 D3) (NAND D4 D0)) (NAND (OR D2
D4) (OR D2 D1))))) (NOR (NOR (OR (NOR (AND D2 D0) (NOR D1 D4))
(AND (NOR (NOR (OR (NOR (AND D2 D0) (NOR D1 D4)) (AND (NOR D0
D2) (NAND D4 D0))) (AND (AND (NAND D4 D3) (OR D3 D0)) (OR D4
D3))) (NOR (NOR (AND (AND D1 D1) (AND D4 D2)) (NAND (NAND D0
D2) (NAND D4 D0))) (NAND (AND D0 D4) (NAND (NOR D1 D4) (OR D1
D0))))) (NAND (AND D4 D1) (OR D2 D0)))) (AND (AND (NAND D4 D3)
(OR D3 D0)) (OR D2 D1))) (NOR (NOR (AND (AND D1 D1) (NOR (NAND
(NAND D4 D2) (NAND D4 D4)) (OR (AND D2 D0) (AND D4 D1))))
(NAND (OR (AND (OR (AND D3 D0) (OR D4 (NAND (OR (NOR D1 D2)
D3) D4))) (NAND (NAND D0 D1) (NAND D2 D2))) (NAND (AND (NOR D0
D1) (OR D3 D4)) (OR (NAND D3 D4) (AND D3 D1)))) (NAND D4 (AND
(OR D1 D0) (OR D3 D2))))) (NAND (OR (NAND D1 D0) (NOR D2 D0))
(NAND D3 D2))))) (NAND (AND (NAND (OR (NOR (NAND (OR D1 D3)
(AND D0 D4)) (NOR (AND D1 D4) (NOR D2 D2))) (AND (OR (NOR D1
D3) (NOR (AND (NOR D2 D2) (NOR (NOR D2 D2) (AND D2 D1))) (AND
(NAND (NAND D4 D0) (NAND (NOR (NAND (NOR D4 D4) (NOR D0 D4))
(OR (AND D2 D3) (AND D4 D1))) (AND (OR (NOR D3 D4) D1) (AND
(OR D1 D0) (OR D3 D2))))) (OR D2 D3)))) (OR (OR D2 D3) (NAND
D3 D0)))) (NAND (AND (NOR (AND D0 D2) (OR D4 D0)) (AND D4 D1))
(OR (AND D1 D4) (NAND (NAND D1 D3) (OR D3 D1))))) (OR (OR D2
D3) (NAND D3 D0))) (AND (OR (NOR D3 D4) D3) (AND (OR D1 D0)
(OR D3 D2)))))
```

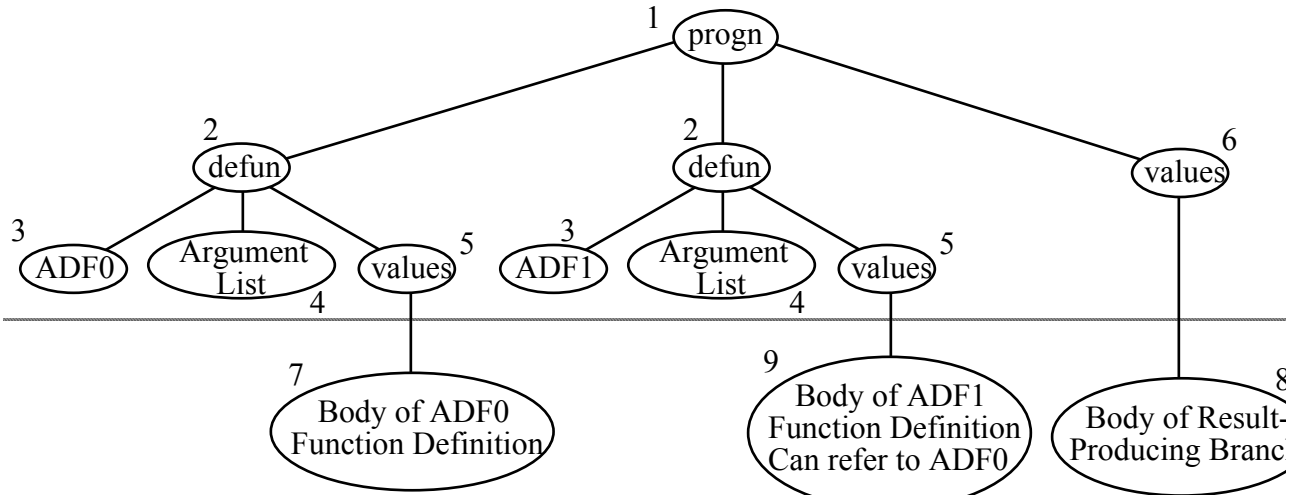
# AUTOMATICALLY DEFINED FUNCTIONS

- **One Result-Producing Branch, RPB**
- **Two function-defining branches (automatically defined functions, ADF)**


PROGN
DEFUN
DEFUN
VALUES
ADF0
(ARG0 ARG1)
ADF1
(ARG0 ARG1 ARG2)
Body of ADF0
Function Definition
Body of ADF1
Function Definition
Result-Producing
Branch

# HIERARCHICAL AUTOMATICALLY DEFINED FUNCTIONS

- **2 ADFs**
- **ADF1 may refer to ADF0**
- **RPB may refer to both ADF0 and ADF1**



# GP TABLEAU WITH ADFS FOR THE EVEN-3-PARITY PROBLEM

| | |
|---|---|
| **Objective:** | **Find a program that produces the value of the Boolean even-3-parity function as its output when given the value of the three independent variables as its input.** |
| **Architecture of the overall program with ADFs:** | **One result-producing branch and two two-argument function-defining branches, with `ADF1` hierarchically referring to `ADF0`.** |
| **Parameters:** | **Branch typing.** |
| **Terminal set for RPB:** | **`D0`, `D1`, and `D2`.** |
| **Function set for RPB:** | **`ADF0`, `ADF1`, `AND`, `OR`, `NAND`, and `NOR`.** |

| | |
|---|---|
| **Terminal set for ADF0:** | **ARG0 and ARG1.** |
| **Function set for ADF0:** | **AND, OR, NAND, and NOR.** |
| **Terminal set for ADF1:** | **ARG0 and ARG1.** |
| **Function set for ADF1:** | **AND, OR, NAND, NOR, and ADF0 (hierarchical reference to ADF0 by ADF1).** |

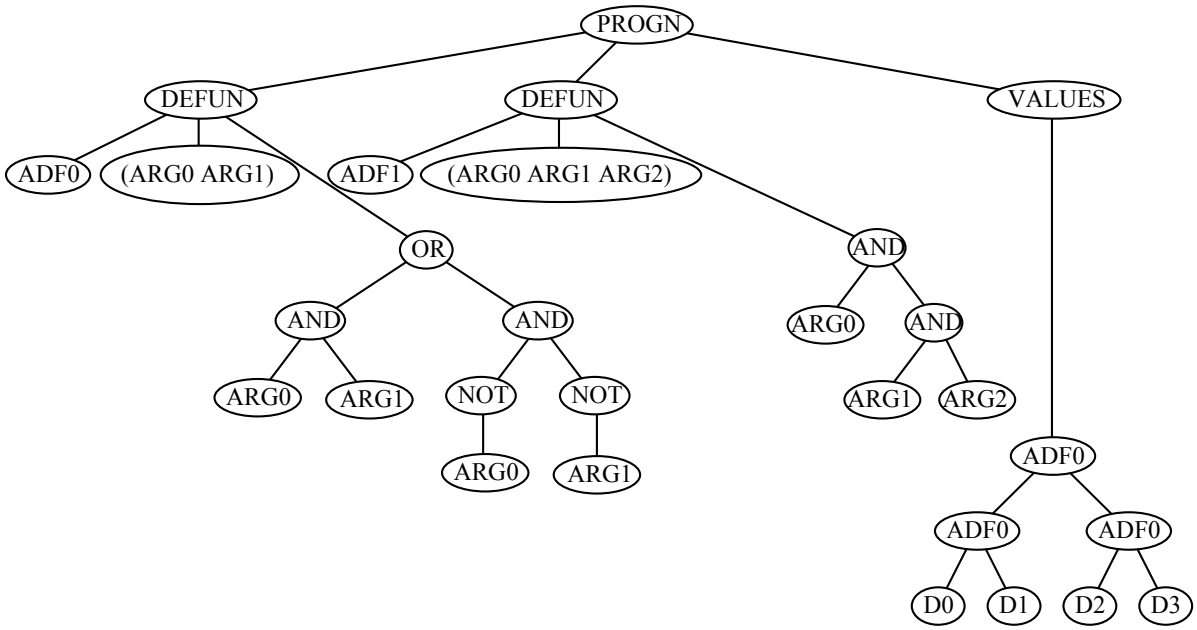
# ILLUSTRATIVE PROGRAM WITH ADF'S FOR EVEN-4-PARITY FUNCTION

- **Automatically Defined Function `ADF0` is even-2-parity (equivalence `EQV`)**
- **Result-Producing Branch `RPB` calls on `ADF0` 3 times**

```
(ADF0 (ADF0 D0 D1) (ADF0 D2
D3))
```

- **`ADF1` is ignored by `RPB`**

# ILLUSTRATIVE PROGRAM WITH ADF'S FOR EVEN-4-PARITY FUNCTION



# EVEN-11-PARITY WITH ADF'S

- **GENERATION 21 - 220 POINTS - 2,048 HITS (OUT OF 2,048)**
- **Automatically Defined Function `ADF0` is**

  **`(EVEN-2-PARITY ARG1 ARG2)`**
- **Automatically Defined Function `ADF1` is**

  **`(EVEN-4-PARITY ARG0 ARG1 ARG2 ARG3)`**

# EVEN-11-PARITY WITH ADF'S

```
(PROGN

        (DEFUN ADF0 (ARG0 ARG1)

             (NAND (NOR (NAND (OR ARG2 ARG1) (NAND ARG1 ARG2))
(NOR (OR ARG1 ARG0) (NAND ARG3 ARG1))) (NAND (NAND (NAND (NAND
ARG1 ARG2) ARG1) (OR ARG3 ARG2)) (NOR (NAND ARG2 ARG3) (OR
ARG1 ARG3)))))

        (DEFUN ADF1 (ARG0 ARG1 ARG2)

             (ADF0 (NAND (OR ARG3 (OR ARG0 ARG0)) (AND (NOR
ARG1 ARG1) (ADF0 ARG1 ARG1 ARG3 ARG3))) (NAND (NAND (ADF0 ARG2
ARG1 ARG0 ARG3) (ADF0 ARG2 ARG3 ARG3 ARG2)) (ADF0 (NAND ARG3
ARG0) (NOR ARG0 ARG1) (AND ARG3 ARG3) (NAND ARG3 ARG0))) (ADF0
(NAND (OR ARG0 ARG0) (ADF0 ARG3 ARG1 ARG2 ARG0)) (ADF0 (NOR
ARG0 ARG0) (NAND ARG0 ARG3) (OR ARG3 ARG2) (ADF0 ARG1 ARG3
ARG0 ARG0)) (NOR (ADF0 ARG2 ARG1 ARG2 ARG0) (NAND ARG3 ARG3))
(AND (AND ARG2 ARG1) (NOR ARG1 ARG2))) (AND (NAND (OR ARG3
ARG2) (NAND ARG3 ARG3)) (OR (NAND ARG3 ARG3) (AND ARG0
ARG0)))))

        (VALUES

             (OR (ADF1 D1 D0 (ADF0 (ADF1 (OR (NAND D1 D7) D1)
(ADF0 D1 D6 D2 D6) (ADF1 D6 D6 D4 D7) (NAND D6 D4)) (ADF1
(ADF0 D9 D3 D2 D6) (OR D10 D1) (ADF1 D3 D4 D6 D7) (ADF0 D10 D8
D9 D5)) (ADF0 (NOR D6 D9) (NAND D1 D10) (ADF0 D10 D5 D3 D5)
(NOR D8 D2)) (OR D6 (NOR D1 D6))) D1) (NOR (NAND D1 D10) (ADF0
(OR (ADF0 D6 D2 D8 D4) (OR D4 D7)) (NOR D10 D6) (NOR D1 D2)
(ADF1 D3 D7 D7 D6))))))
```

# SIMPLIFIED SOLUTION FROM GENERATION 21 FOR EVEN-11-PARITY WITH ADF'S

```
(OR (EVEN-4-PARITY D1 D0 (EVEN-
2-PARITY (EVEN-4-PARITY (EVEN-
2-PARITY D3 D2) (OR D10 D1)
(EVEN-4-PARITY D3 D4 D6 D7)
(EVEN-2-PARITY D8 D9)) (EVEN-2-
PARITY (NAND D1 D10) (EVEN-2-
PARITY D5 D3))) D1) (NOR (NAND
D1 D10) (EVEN-2-PARITY (NOR D10
D6) (NOR D1 D2))))
```

# SIMPLIFIED SOLUTION FROM GENERATION 21 FOR EVEN-11-PARITY WITH ADF'S