Genetic Programming

Companion slides of
Genetic Programming. On the Programming of Computers by Means of Natural Selection
by John R. Koza
MIT Press, Six Edition, 1998

Overview of Genetic Programming

Genetic Algorithms

+

increasing complexity of the structures undergoing adaptation

Structures = hierarchical computer programs of dynamically varying size and shape

Overview of Genetic Programming

Search Space = possible computer programs
for the fittest individual computer program

Overview of Genetic Programming

initial population

=

randomly generated computer programs
composed of functions and terminals

Overview of Genetic Programming

each individual (computer program)

is

measured in terms of how well it performs in the particular problem environment (fitness measure)

Overview of Genetic Programming

each individual (computer program)

is

run over a number of different fitness cases

fitness = measured as a sum or an average!

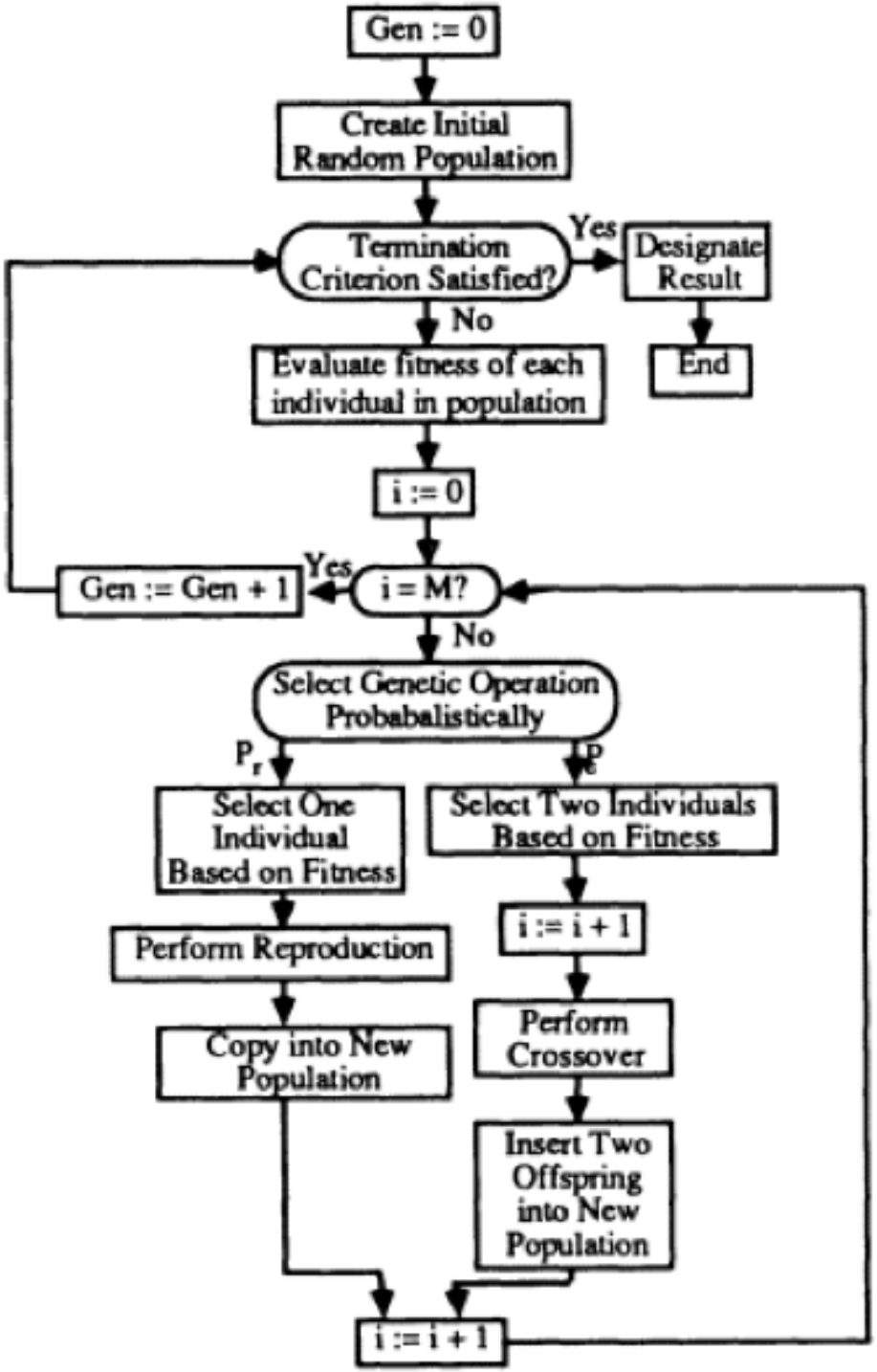
Overview of Genetic Programming

size & shape of the solution is not specified

Overview of Genetic Programming

preprocessing & postprocessing

do not exist



Flowchart of Genetic Programming

Overview of Genetic Programming

1. The Structures Undergoing Adaptation
2. Initial Structures
3. Fitness
4. Operations for Modifying Structures
5. States of the Adaptive Systems
6. Termination Criterion
7. Designating a Result, and the parameters that control the process

The Structures Undergoing Adaptation

The Structures Undergoing Adaptation

= a population of individuals

Individual

= hierarchical structured computer programs whose size, shape and contents can dynamically change during the process

The Structures Undergoing Adaptation

Individual = composition between

$F = \{f_1, f_2, ..., f_{Nfunc}\}$, $arity(f_i) = z(f_i)$

and

$T = \{a_1, a_2, ..., a_{Nterm}\}$

Function Set

arithmetic operations (+, -, *, etc.)

mathematical functions (sin, cos, exp, log)

Boolean operations (AND, OR, NOT)

conditional operators (If-Then-Else)

functions causing iteration (Do-Until)

functions causing recursion

other domain-specific functions

Terminal Set

variable atoms

constant atoms

functions taking no explicit arguments

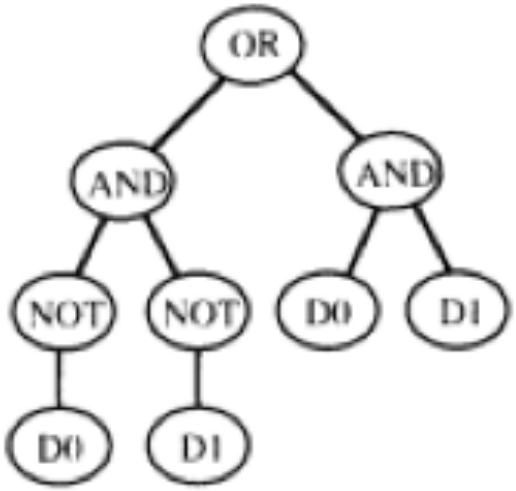
Example

F = {AND, OR, NOT}

T = {D0, D1}

C = F ∪ T = {AND, OR, NOT, D0,D1}

Example: even-2-parity function



LISP S-expression

(OR (AND (NOT D0) (NOT D1)) (AND D0 D1))

Search Space

the space of all possible LISP S-expressions that can be recursively created by compositions of the available functions and available terminals for the problem.

Closure of the Function Set and Terminal Set

closure property

=

each function is able to accept as its arguments any value and data type that may possibly be returned by any function set and any value and data type that may possibly be assumed by any terminal

Closure Property: Arithmetic Operation of Division

```
(defun % (numerator denominator)
"The Protected Division Function"
(if (= 0 denominator) 1 (/ numerator denominator)))
```

Closure Property: Arithmetic Operation of Division

```
(defun % (numerator denominator)
"The Protected Division Function"
(if (= 0 denominator) 1 (/ numerator denominator)))
```

or return

:undefined

Closure Property: Square Root

```
(defun srt (argument)
"The Protected Square Root Function"
(sqrt (abs argument)))
```

Closure Property: Natural Logarithm

```
(defun rlog (argument)
"The Protected Natural Logaritm Function"
(if (= 0 argument) 0 (log (abs argument))))
```

Numerical-valued logic

```
(defun gt (first second)
"The numerically-valued GT Function"
(if (> 0 first second) 1 -1)))
```

Conditional comparative operators can be redefined

(+ (* 3 4) 5)

Conditional comparative operators can be redefined

(+ (* 3 4) 5)

* 3 4 may have a side effect

Conditional comparative operators can be redefined

iflth (If Less Than Zero)

suppress premature evaluation of the arguments

define a LISP macro

Conditional branching operators can be redefined

execute an alternative depending on an external state or condition

(IF-FOOD-AHEAD (MOVE) (TURN-RIGHT))

Sufficiency of the Function Set and Terminal Set

sufficiency property

=

the set of terminals and primitive functions are able to express a solution to the problem

Boolean Functions

F = {AND, OR, NOT}

F = {AND, NOT}

Universality of Selecting Primitive Functions and Terminals

steps

=

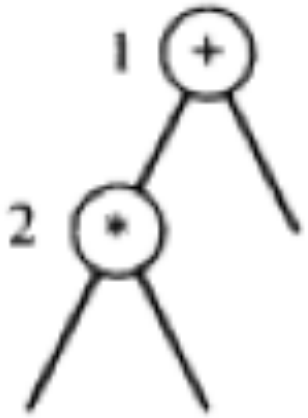
similar steps in other machine learning paradigms

The Initial Structures

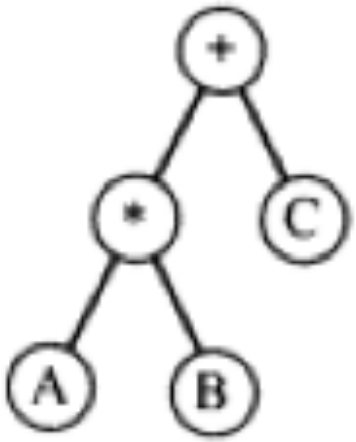
generate randomly a rooted tree with other branches representing the S-expression

The Initial Structures

The Initial Structures



The Initial Structures



The Initial Structures

- full method
- grow method
- ramped half-and-half

Ramped half-and-half

MAX_DEPTH = 6

MIN_DEPTH = 2

20% - depth i, i $\in$ {2,3,4,5,6}

50% created with full method

50% created with grow method

Variety of population

Variety

=

percentages of individuals for which no exact duplicate exists elsewhere in the populations

Fitness

- Raw Fitness
- Standardized Fitness
- Adjusted Fitness
- Normalized Fitness

Raw Fitness

is the measurement of fitness that is stated in the natural terminology of the problem itself

Standardized Fitness

a lower numerical value is always a better value

=

maximum raw fitness - observed raw fitness

Adjusted Fitness

$$a(i,t) = \frac{1}{1 + s(i,t)},$$

Adjusted Fitness

standardized fitness $\in \{1,..,64\}$

64 → af = 0.0154

63 → af = 0.0159

4 → af = 0.20

3 → af = 0.25

Normalized Fitness

- ranges between 0 and 1
- is larger for better individuals
- the sum of the normalized fitness value is 1

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{M} a(k,t)}.$$

Greedy Over-Selection

over selecting the fitter individuals in the population for M > 500

Primary Operations for Modifying Structures

- Selection
- Crossover

Selection

select a single S-expression

copy from the current population into the new population

Crossover

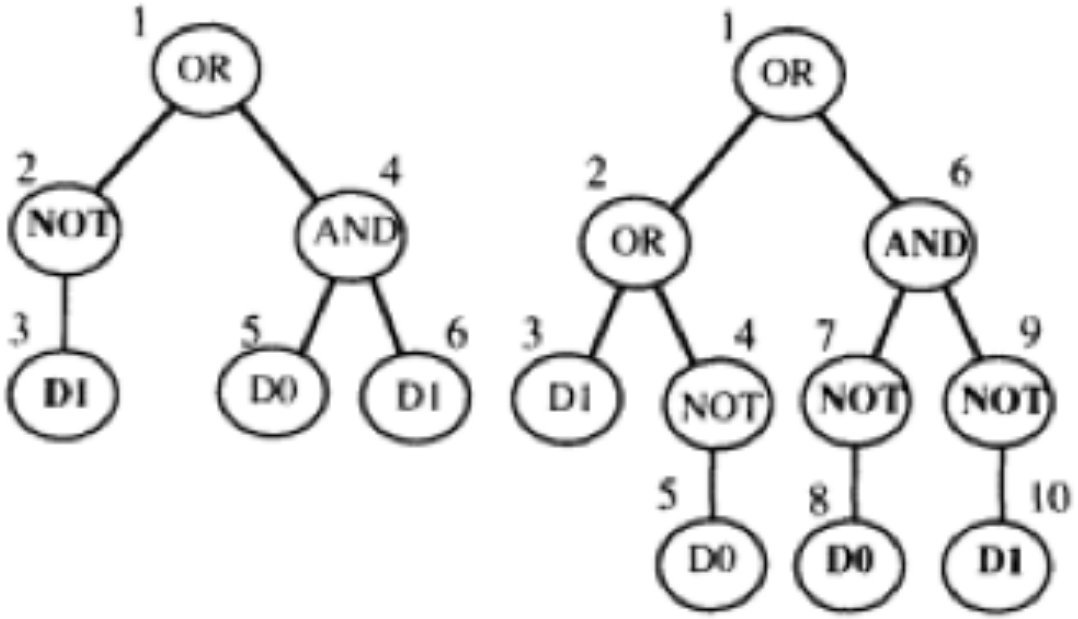
(OR **(NOT D1)** (AND D0 D1))

(OR

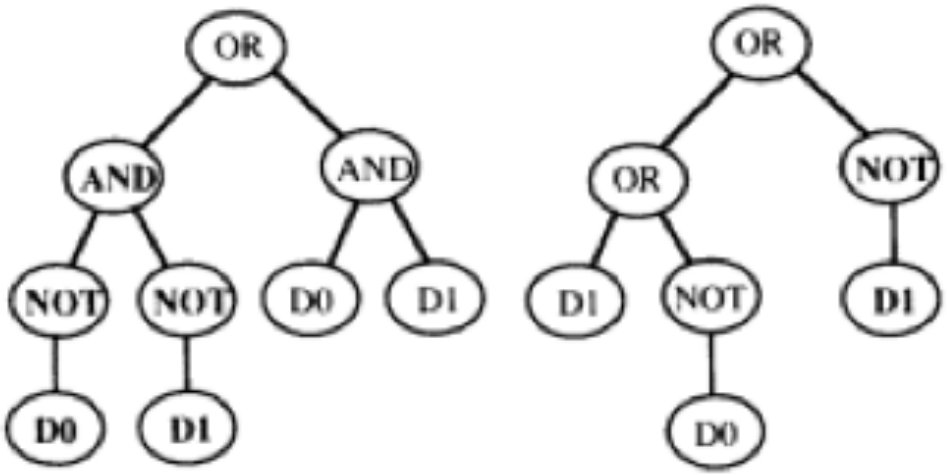
(OR D1 (NOT D0))

(AND (NOT D0) (NOT D1)))

Crossover


1
OR
2
NOT
4
AND
3
D1
5
D0
6
D1
1
OR
2
OR
6
AND
3
D1
4
NOT
7
NOT
9
NOT
5
D0
8
D0
10
D1

Crossover



Secondary Operations for Modifying Structures

- Mutation
- Permutation
- Editing
- Encapsulation
- Decimation

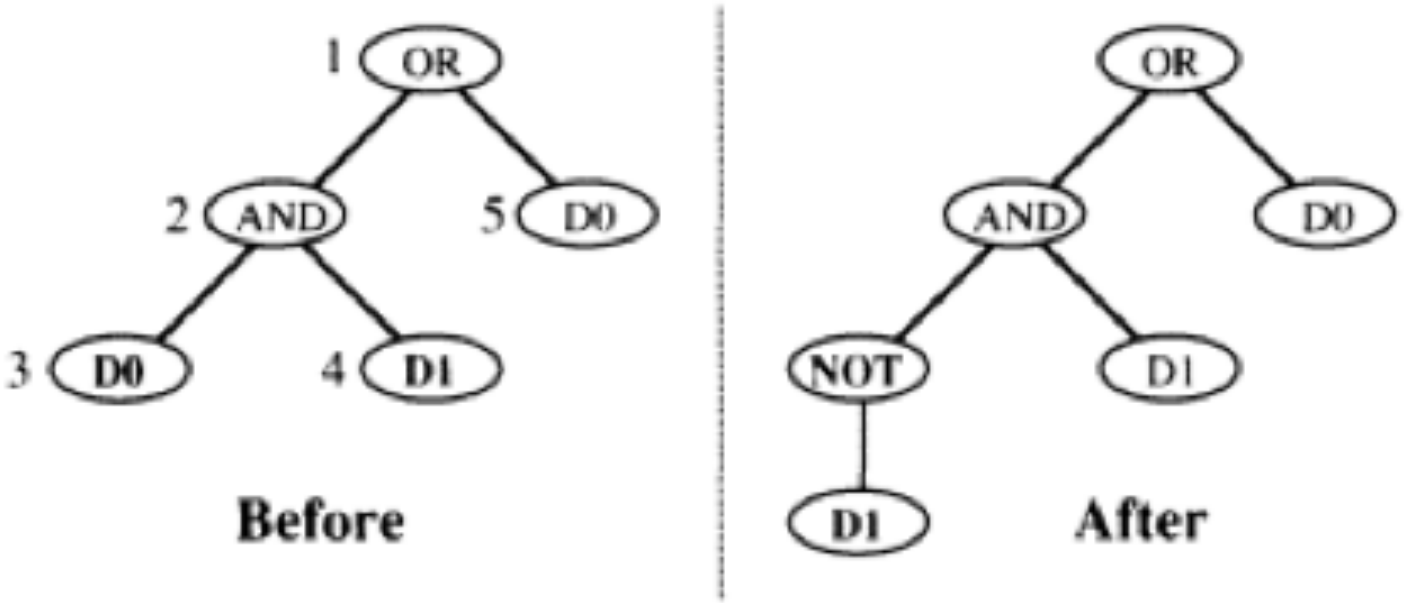
Mutation

selects a random point (function or terminal)

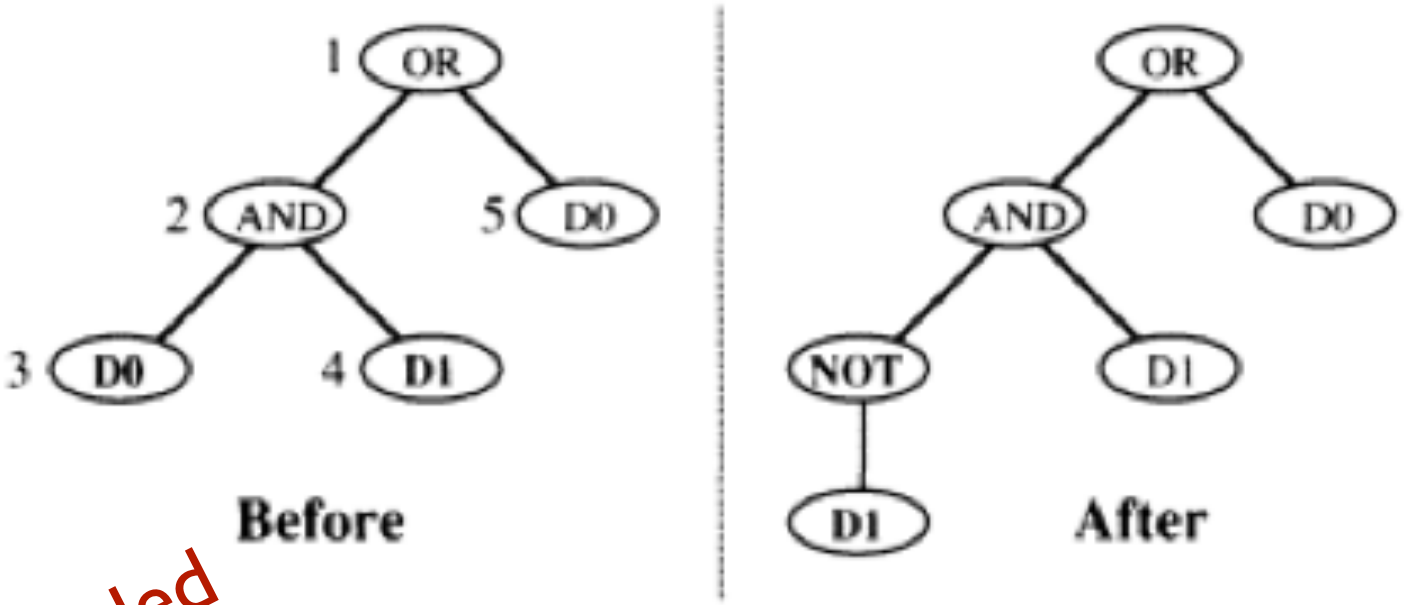
removes selected point and whatever is below it

inserts a randomly generated subtree at that point

Mutation

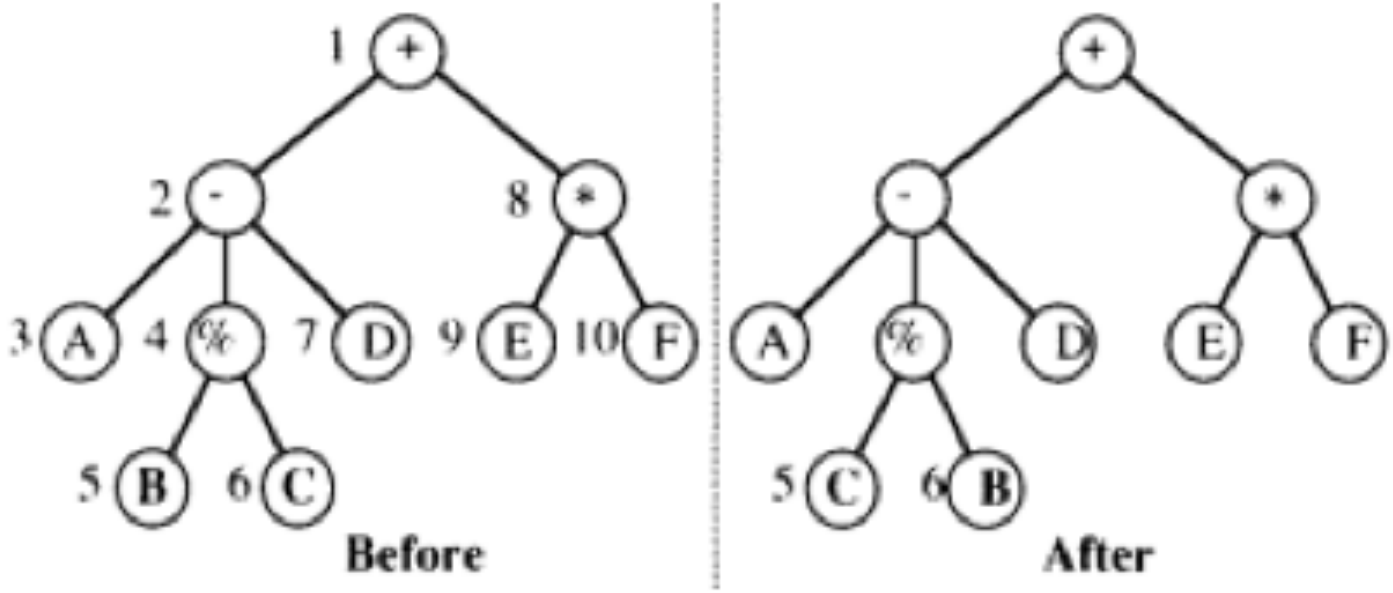


Mutation



Not needed in Genetic Programming

Permutation



Editing

provides a means to edit and simplify

S-expressions as genetic programming is running according to domain-specific editing rules

Editing

(+ 1 2) → 3

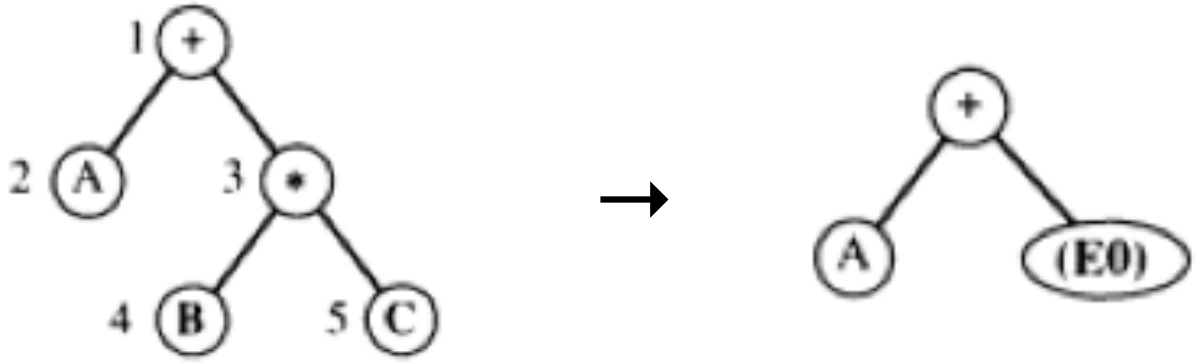
(AND T T) → T

(AND X X) → X

(OR X X) → X

(NOT (NOT X)) → X

Encapsulation



Decimation

fitness may be skewed

keep in the initial population (e.g. 10%) only the individuals performing good

State of the Adaptive System

- current population
- control parameters, terminal and function set, best-so-far individual

Termination Criterion

- G generations have been run
- find the correct solution, ...

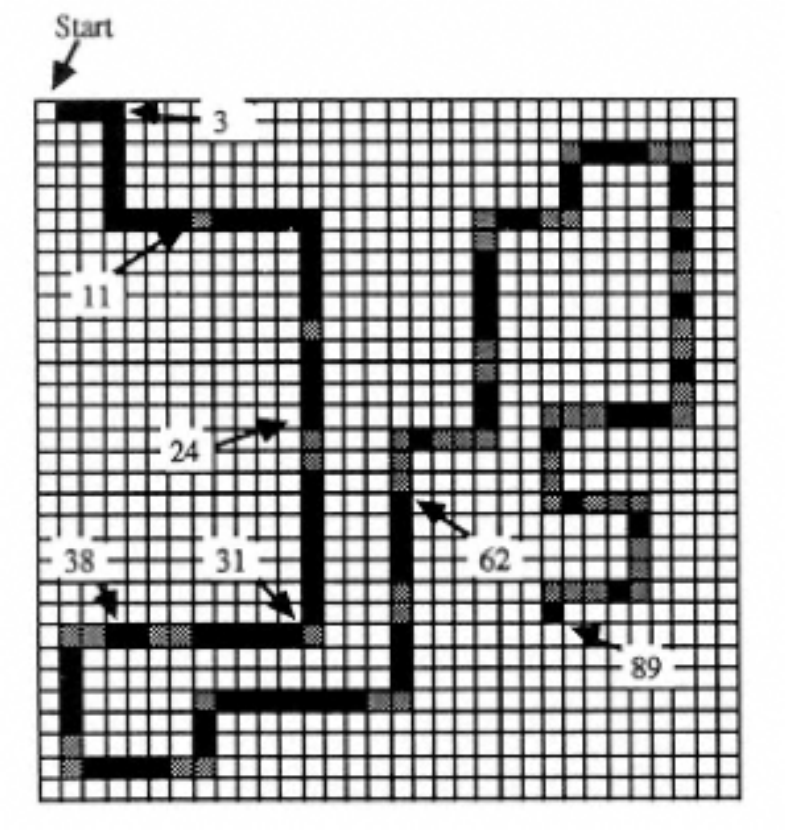
Some Examples of Genetic Programming

Five ingredients

- Set of Terminals
- Set of Functions
- How is fitness measured?
- Parameters / variables for controlling the run
- Result and criterion for terminating a run

Artificial Ant

The Santa Fe Trail



89 Food Pellets

single gaps

double gaps

single gap at corners

double gap at corners

triple gap at corners

Artificial Ant's Characteristics

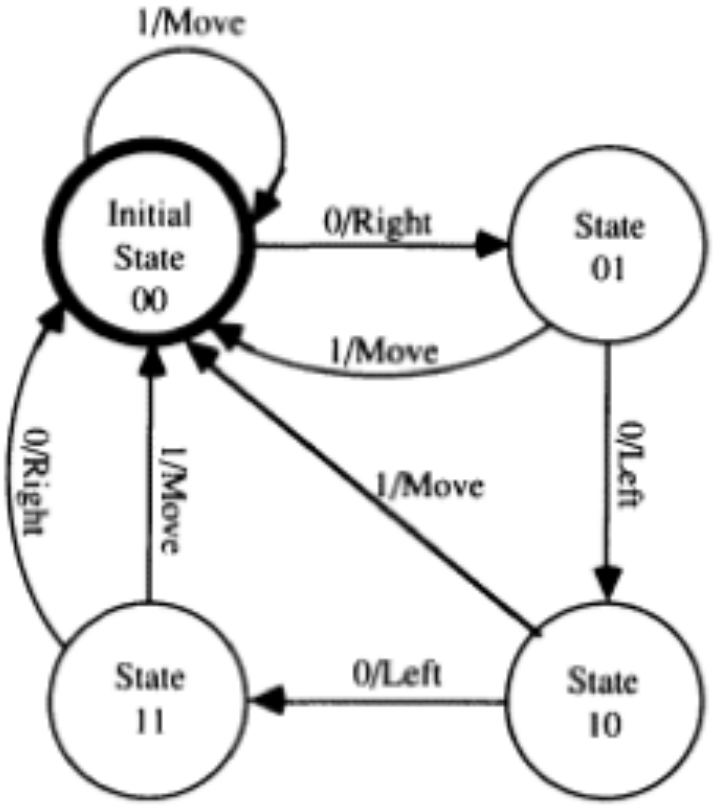
sensor for smelling the food

RIGHT, LEFT (without moving the ant)

MOVE in the direction it is currently facing

NO-OP

An Individual when using Genetic Algorithm



Terminals Set

T = {(MOVE), (RIGHT), (LEFT)}

Functions Set

F = {

IF-FOOD-AHEAD

(PROGN2 P1 P2)

(PROGN3 P1 P2 P3)

}

Fitness

Raw Fitness = the amount of eaten food

Standardized Fitness = 89 - Raw Fitness

Table 7.3 Tableau for the artificial ant problem for the Santa Fe trail.	
Objective:	Find a computer program to control an artificial ant so that it can find all 89 pieces of food located on the Santa Fe trail.
Terminal set:	`(LEFT), (RIGHT), (MOVE).`
Function set:	`IF-FOOD-AHEAD, PROGN2, PROGN3.`
Fitness cases:	One fitness case.
Raw fitness:	Number of pieces of food picked up before the ant times out with 400 operations.
Standardized fitness:	Total number of pieces of food (i.e., 89) minus raw fitness.
Hits:	Same as raw fitness for this problem.
Wrapper:	None.
Parameters:	$M = 500$. $G = 51$.
Success predicate:	An S-expression scores 89 hits.

Some Initial Generated Individuals

(PROGN2 (RIGHT) (LEFT))

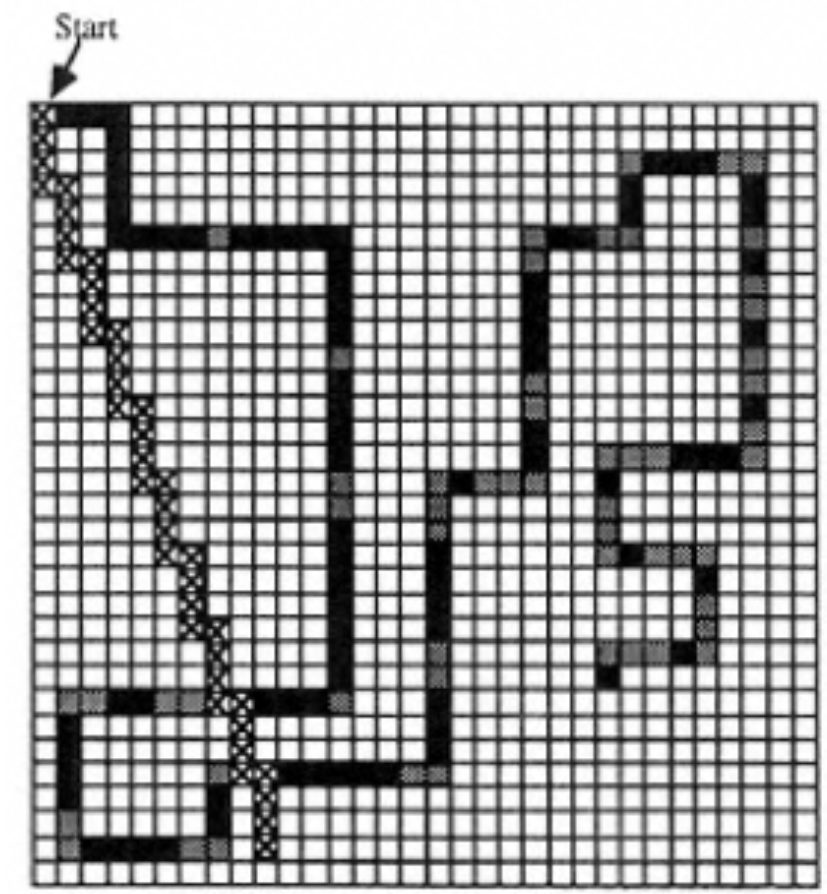
(IF-FOOD-AHEAD (RIGHT) (LEFT))

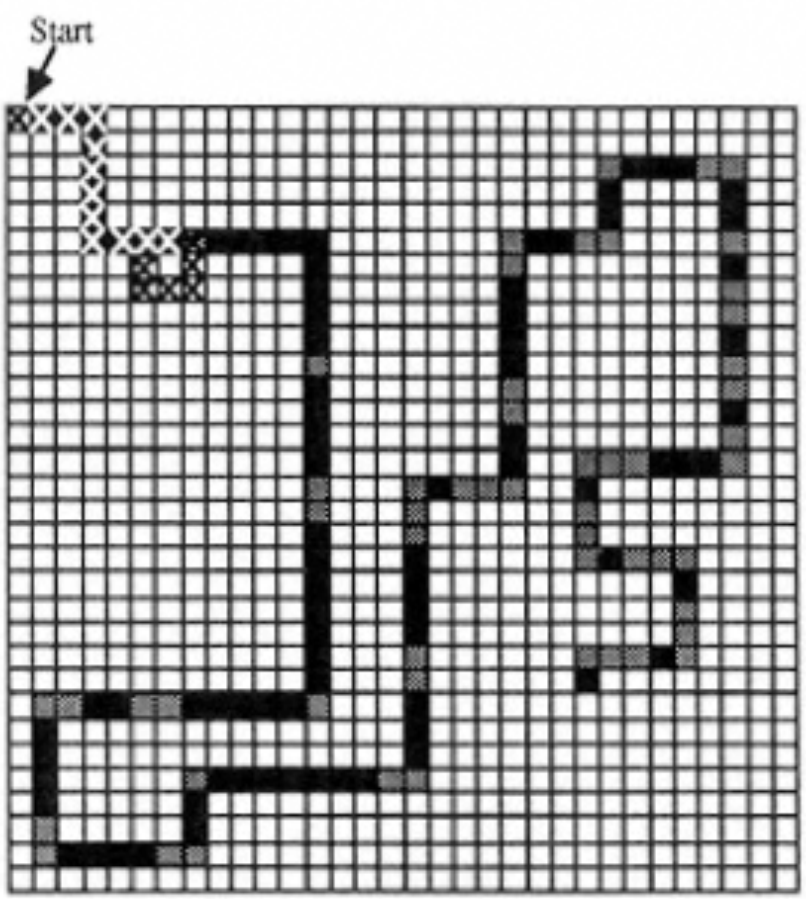
The Ant does not move

Some Initial Generated Individuals

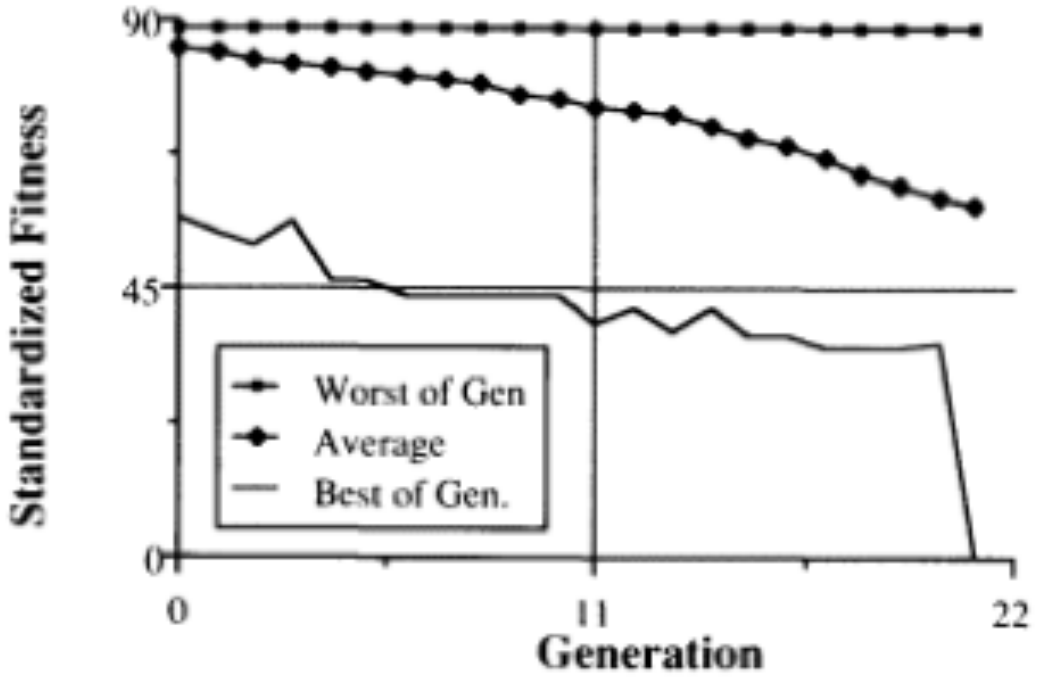
(PROGN2 (MOVE) (MOVE))

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

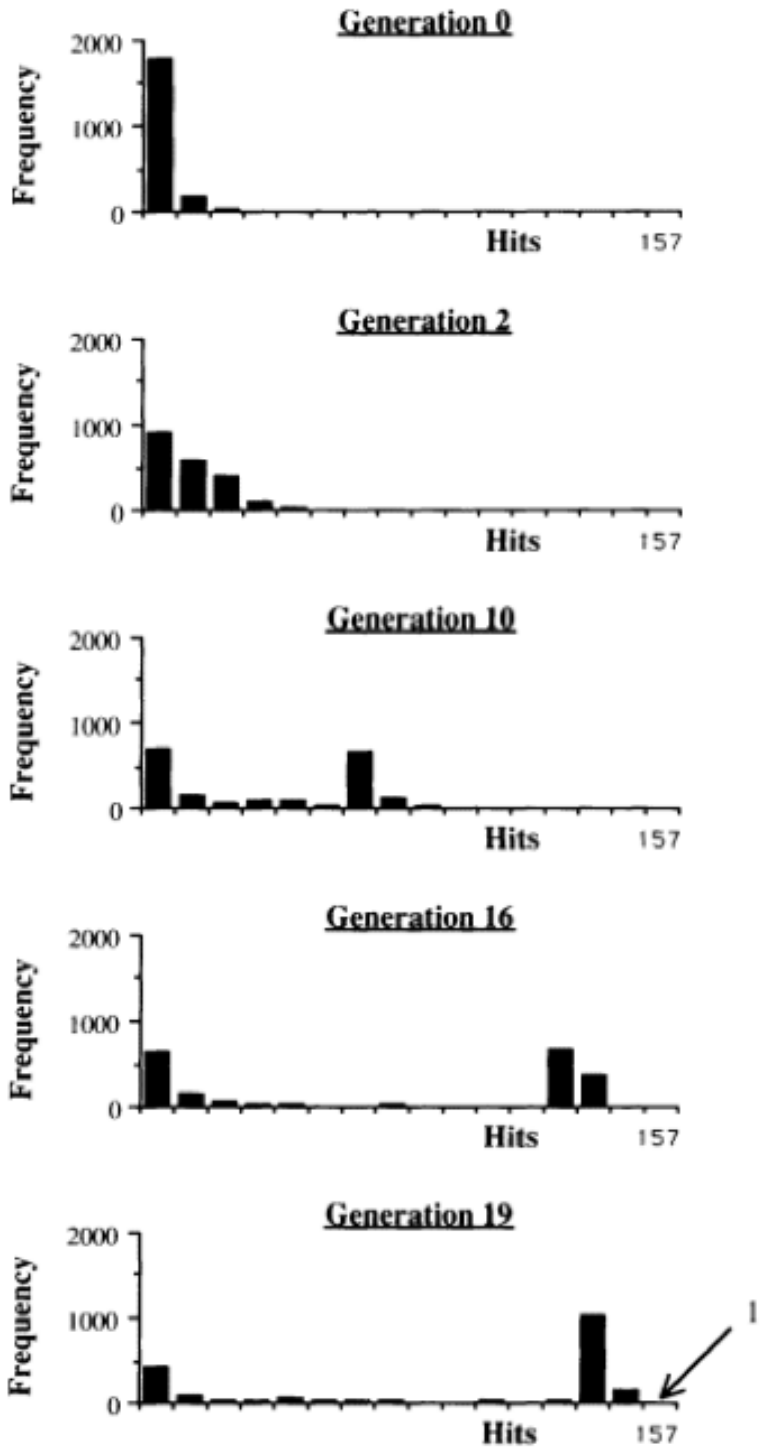

Start


Start

After Selection and Crossover . . .



Best-of-run individual . . .



The Los Altos Hills Trail

The Santa Fe Trail

+

two steps to the left or
two steps to the right

Simple Symbolic Regression

Linear Regression

Independent Variable(s)

?

Dependent Variable(s)

find numerical coefficients of a linear combination of the independent variable(s)

Quadratic Regression

Independent Variable(s)

?

Dependent Variable(s)

find numerical coefficients of a quadratic expression of the independent variable(s)

Fourier Regression

Independent Variable(s)

?

Dependent Variable(s)

find numerical coefficients of a harmonic expression of the independent variable(s)

?

discovery both

- the correct functional form that fits the data
- the appropriate numeric coefficients that go with that functional form

Sample

20 pairs of points $\in$ [-1.0, +1.0]

(x_i, y_i)

Terminals and Functions Set

T = {X}

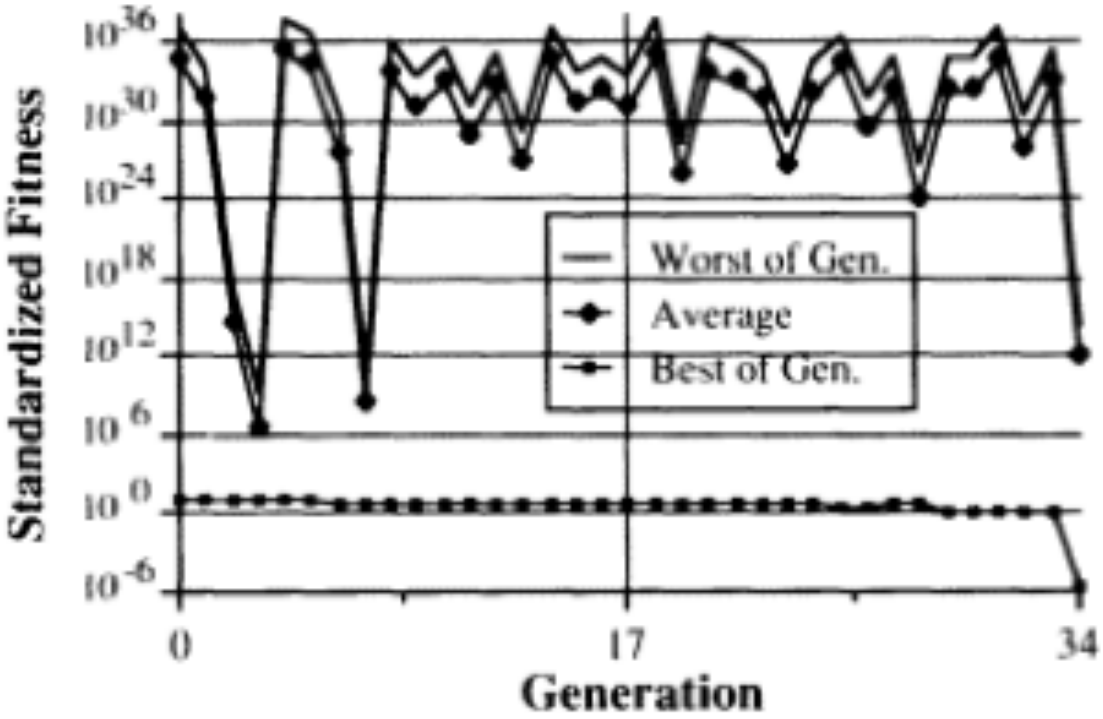
F = {+, -, *, %, SIN, COS, EXP, RLOG}

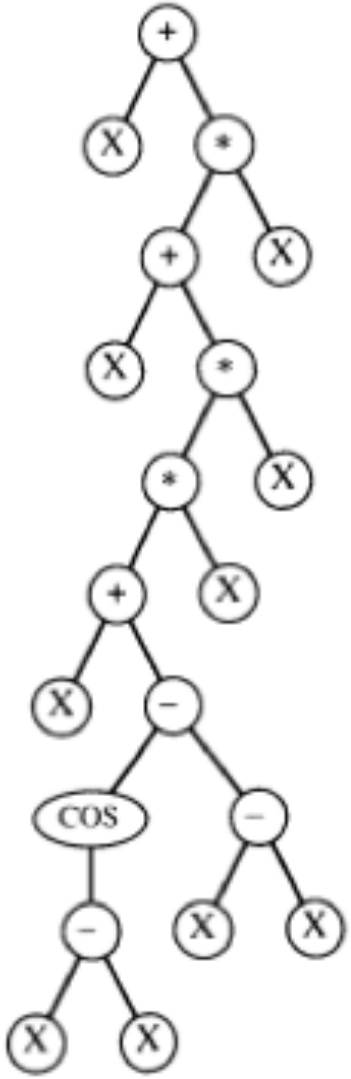
Fitness

Raw Fitness = the sum, taken over the 20 fitness cases, of the absolute value of the difference (error) between the value in the real-valued ranged space produce by the S-expression for a given value of the independent variable x_i and the correct y_i in the range space.

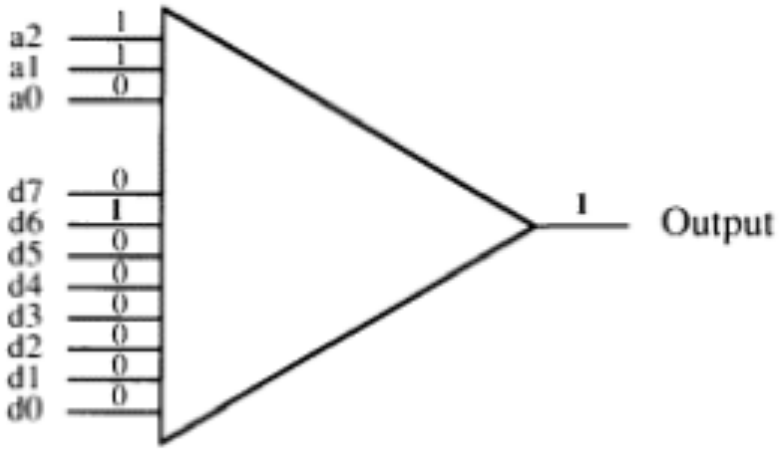
Raw Fitness of Initial Population

Worst-of-generation individual	1038
Median individual	23,67
Second-best individual	6,05
Best-of-generation individual	4,47


Standardized Fitness
10^{36}
10^{30}
10^{24}
10^{18}
10^{12}
10^{6}
10^{0}
10^{-6}
Worst of Gen.
Average
Best of Gen.
0
17
34
Generation


+
X
*
+
X
X
*
*
X
+
X
X
−
COS
−
−
X
X
X
X

Boolean 11-Multiplexer



k address bits a_i

2^k data bits

$N = k + 2^k$

Terminals and Functions Set

T = {A0, A1, A2, D0, D1, ..., D7}

F = {AND, OR, NOT, IF}

Search Space

$N = k + 2^k \quad k = 3 \quad N = 11$

11 arguments $\Rightarrow 2^{11} = 2018$ combinations

one function $2^{k+2k} = 2^{2018} \simeq 10^{616}$

Fitness

Raw Fitness = the number of fitness cases (over all 2048) where the Boolean returned value is correct

Standardized Fitness =

MAX(Raw Fitness) - Observed(Raw Fitness)

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.

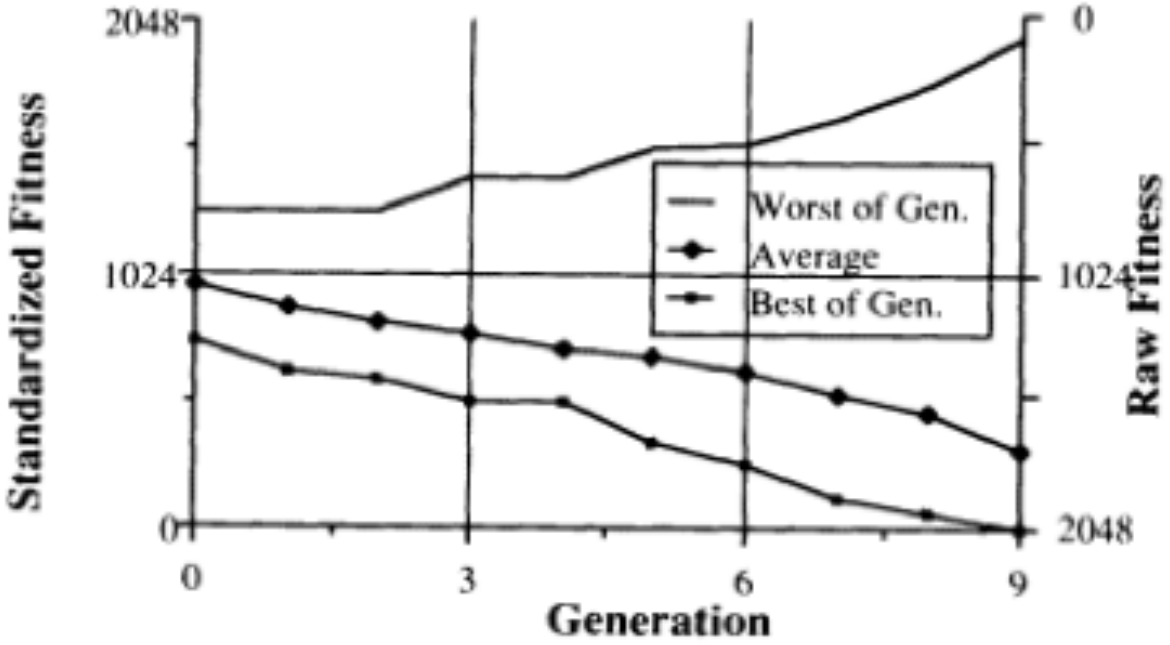
Some Initial Generated Individuals

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

Standardized Fitness of Initial Population

Worst-of-generation individual	1280
23 Best-of-generation individuals	768

One Best-of-generation sample (IF A0 D1 D2)


2048
1024
0
Standardized Fitness
0
1024
2048
Raw Fitness
Worst of Gen.
Average
Best of Gen.
0
3
6
9
Generation

Best-of-generation 9 individual