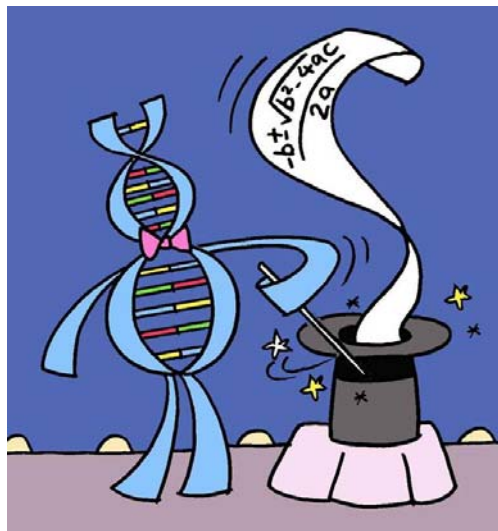


AUTOMATED DESIGN OF COMPLEX STRUCTURES USING DARWINIAN EVOLUTION AND GENETIC PROGRAMMING

EE 380—FEBRUARY 18, 2009



John R. Koza

Consulting Professor

Department of Electrical Engineering

Stanford University

koza@stanford.edu

<http://www.genetic-programming.com/johnkoza.html>

OUTLINE

- **Introduction**
- **The Genetic Programming (GP) algorithm**
- **Developmental GP**
- **Routine human-competitive results**
- **Cross-domain observations about GP**
- **Parallel computing**
- **Qualitative progression of results (Moore's law)**
- **The Future**
- **Sources of additional information**

CHARLES DARWIN

“I think it would be a most extraordinary fact if no variation ever had occurred useful to each being's own welfare

“But if variations useful to any organic being do occur, assuredly individuals thus characterised will have the best chance of being preserved in the struggle for life; and from the strong principle of inheritance they will tend to produce offspring similarly characterised.

“This principle of preservation, I have called, for the sake of brevity, *Natural Selection*.”

— Charles Darwin, *On the Origin of Species by Means of Natural Selection* (1859)

TURING'S THREE APPROACHES TO MACHINE INTELLIGENCE

- **Turing made the connection between searches and the challenge of getting a computer to solve a problem without explicitly programming it in his 1948 essay “Intelligent Machines”**

“Further research into intelligence of machinery will probably be very greatly concerned with 'searches' ...”

TURING

1. LOGIC-BASED SEARCH

One approach that Turing identified is a search through the space of integers representing candidate computer programs.

2. "CULTURAL SEARCH"

Another approach is the "cultural search" which relies on knowledge and expertise acquired over a period of years from others (akin to present-day knowledge-based systems).

3. "GENETICAL OR EVOLUTIONARY SEARCH"

"There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value."

TURING

"We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications"

"Structure of the child machine = Hereditary material"

"Changes of the child machine = Mutations"

"Natural selection = Judgment of the experimenter"

— Turing's 1950 paper "Computing Machinery and Intelligence"

REASON FOR GENETIC PROGRAMMING

THE CHALLENGE

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

— Attributed to Arthur Samuel (1959)

CRITERION FOR SUCCESS

"The aim [is] ... to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence."

— Arthur Samuel (1983)

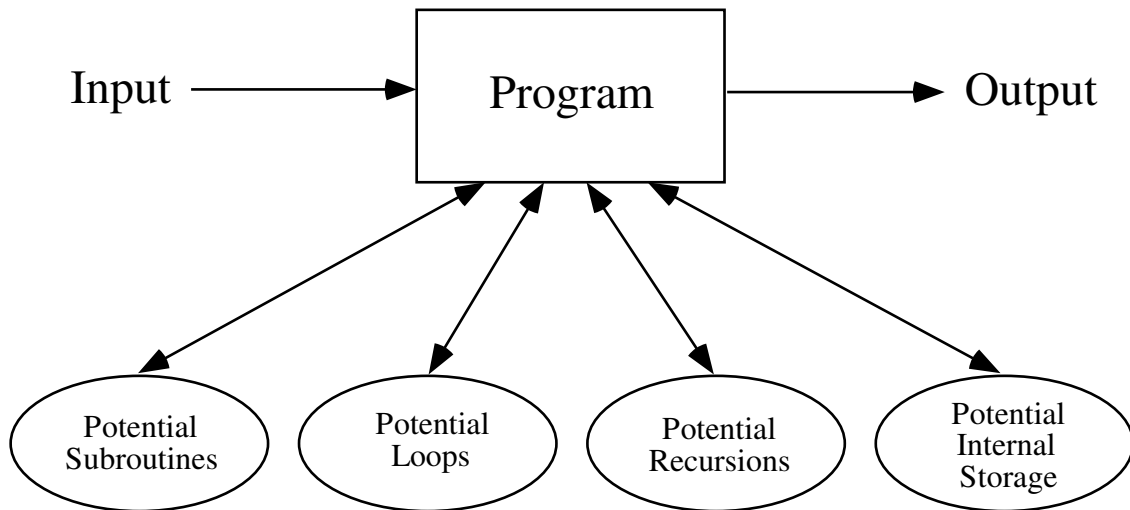
VARIOUS REPRESENTATIONS USED TO TRY TO ACHIEVE ARTIFICIAL INTELLIGENCE (AI) AND MACHINE LEARNING (ML)

- **Decision trees**
- **If-then production rules (e.g., expert systems)**
- **Horn clauses**
- **Neural nets (matrices of numerical weights)**
- **Bayesian networks**
- **Frames**
- **Propositional logic**
- **Binary decision diagrams**
- **Formal grammars**
- **Numerical coefficients for polynomials**
- **Tables of values (reinforcement learning)**
- **Conceptual clusters**
- **Concept sets**
- **Parallel if-then rules (e.g., learning classifier systems)**

REPRESENTATION

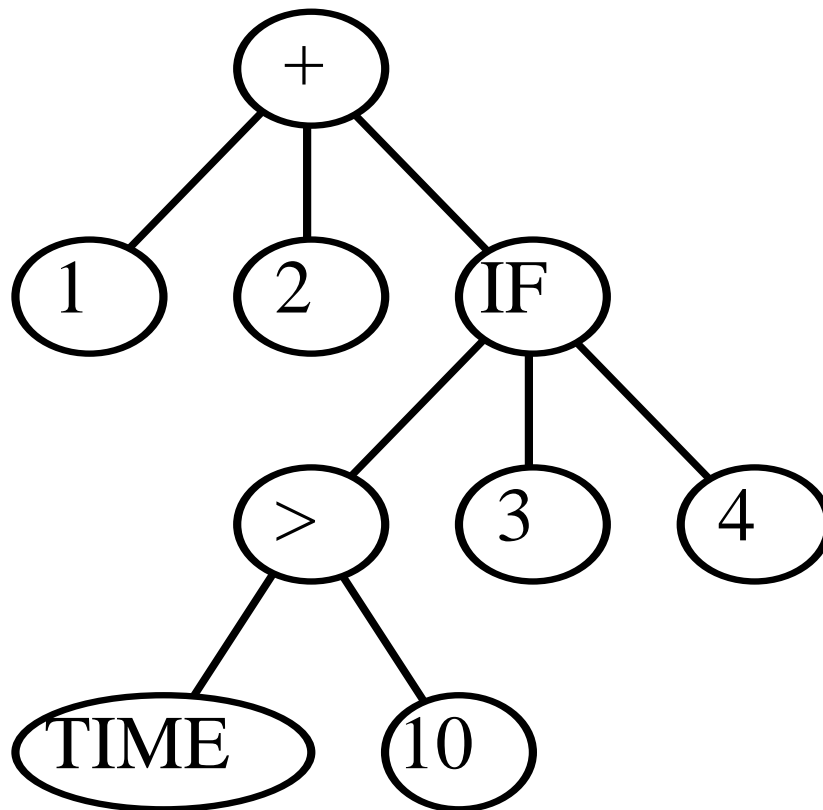
- **“Our view is that computer programs are the best representation of computer programs.”**

A COMPUTER PROGRAM



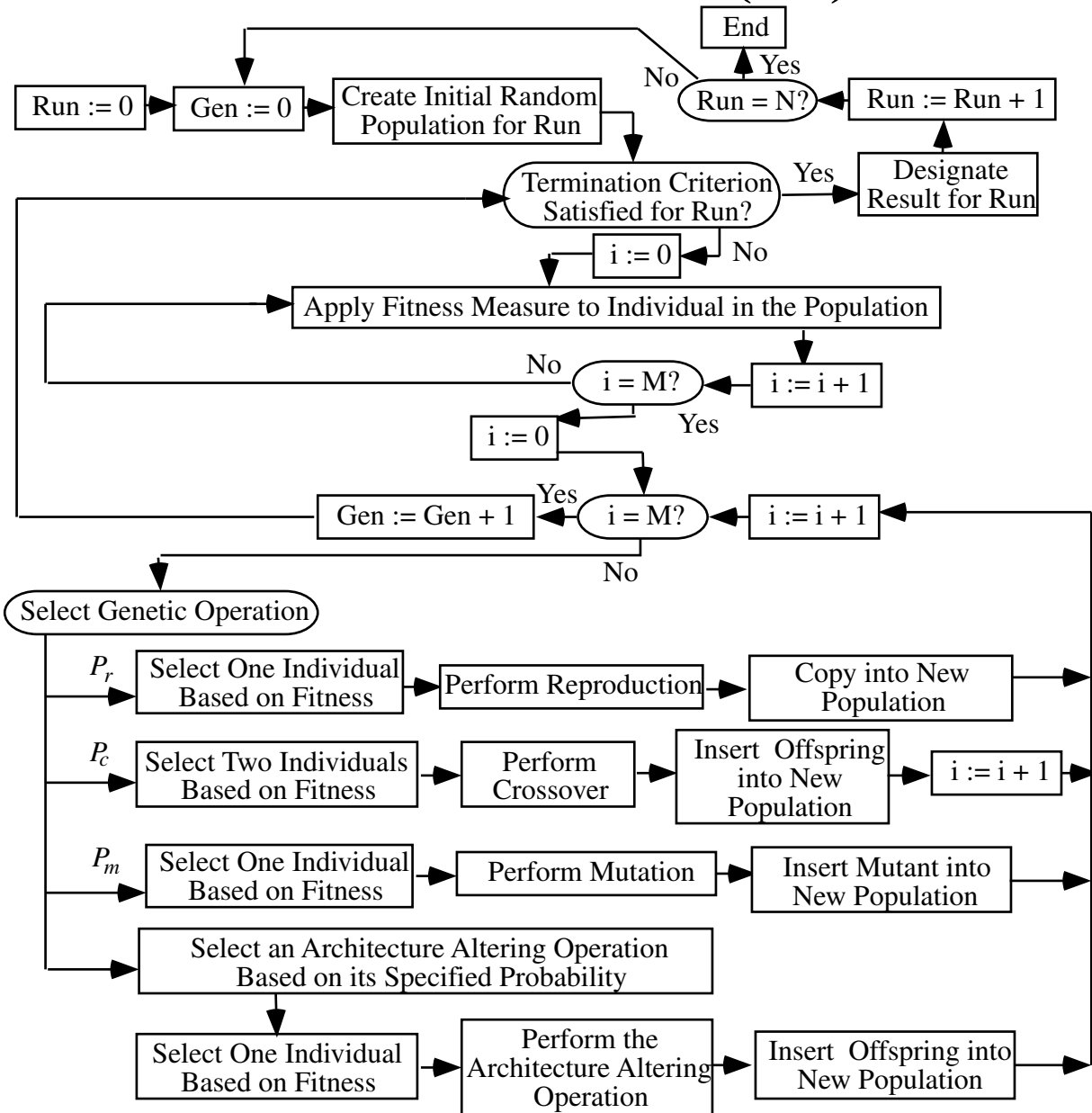
**COMPUTER PROGRAM
= PARSE TREE = PROGRAM TREE
= PROGRAM IN LISP = DATA = LIST**

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

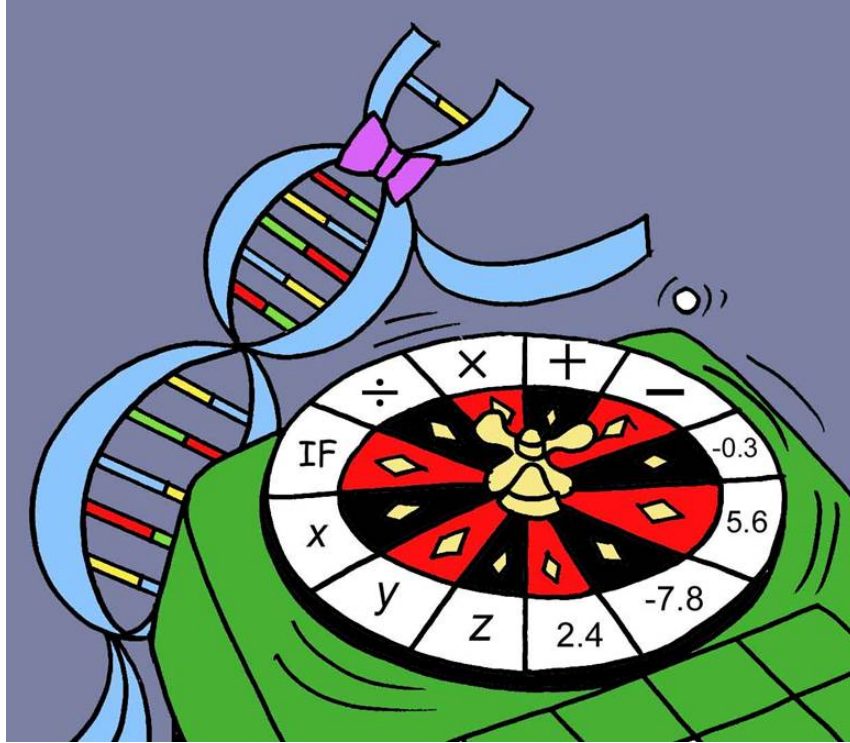


- Terminal set $T = \{1, 2, 10, 3, 4, \text{TIME}\}$
- Function set $F = \{+, \text{IF}, >\}$

FLOWCHART FOR GENETIC PROGRAMMING (GP)



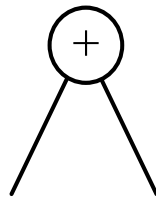
RANDOM CREATION OF A PROGRAM TREE



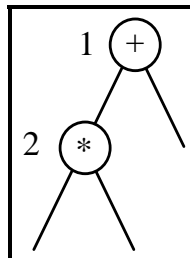
RANDOM CREATION OF A PROGRAM TREE

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

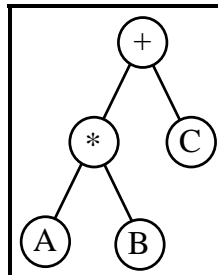
BEGIN WITH TWO-ARGUMENT +



CONTINUE WITH TWO-ARGUMENT *



FINISH WITH TERMINALS A, B, AND C

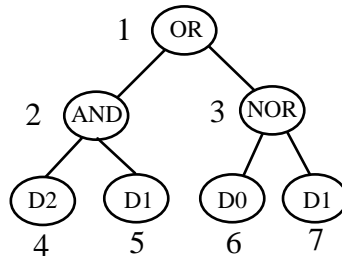


- The result is a syntactically valid executable program (provided the set of functions is “closed”)

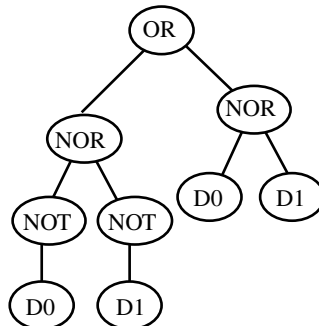
MUTATION OPERATION

- Select parent probabilistically based on fitness
- Pick point from 1 to NUMBER-OF-POINTS
- Delete subtree at the picked point
- Grow new subtree at the mutation point in same way as generated trees for initial random population (generation 0)
- The result is a syntactically valid executable program

ONE PARENTAL PROGRAM



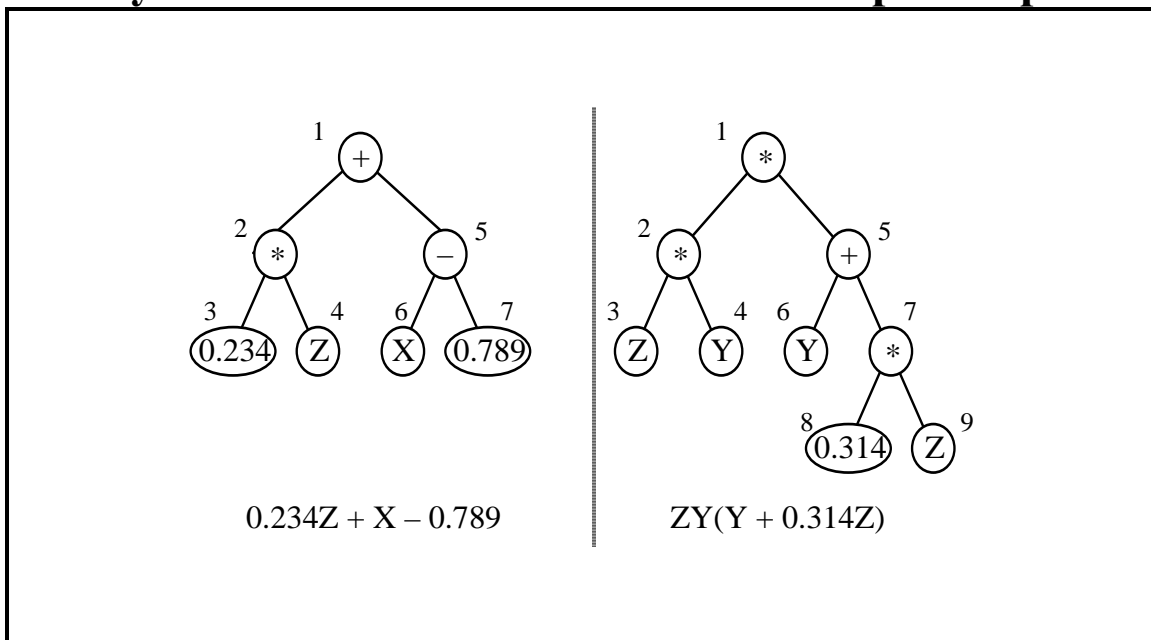
OFFSPRING PRODUCED BY MUTATION



- The result is a syntactically valid executable program

CROSSOVER (SEXUAL RECOMBINATION) OPERATION FOR COMPUTER PROGRAMS

- Select two parents probabilistically based on fitness
- Randomly pick a number from 1 to NUMBER-OF-POINTS – independently for each of the two parental programs
- Identify the two subtrees rooted at the two picked points



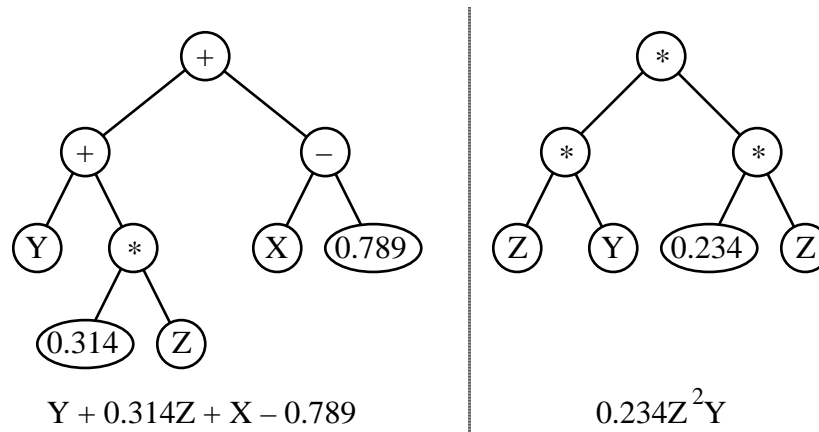
Parent 1:

$(+ \ (\underline{* \ 0.234 \ Z}) \ (- \ X \ 0.789))$

Parent 2:

$(* \ (* \ Z \ Y) \ \underline{(+ \ Y \ (* \ 0.314 \ Z)})$

THE CROSSOVER OPERATION (TWO OFFSPRING VERSION)



Offspring 1:

$$\left(+ \frac{\left(+ Y \left(* 0.314 Z \right) \right)}{\left(- X 0.789 \right)} \right)$$

Offspring 2:

$$\left(* \left(* Z Y \right) \frac{\left(* 0.234 Z \right)}{\left(* 0.234 Z \right)} \right)$$

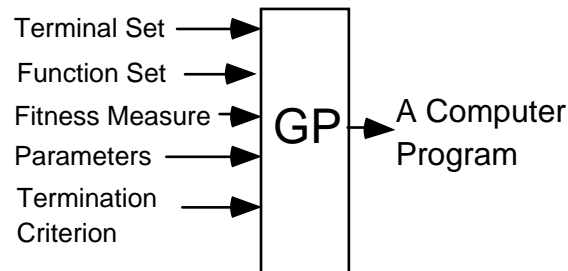
- The result is a syntactically valid executable program

FIVE MAJOR PREPARATORY STEPS FOR GP



FIVE MAJOR PREPARATORY STEPS FOR GP

- **Determining the set of terminals**
- **Determining the set of functions**
- **Determining the fitness measure**
- **Determining the parameters for the run**
- **Determining the method for designating a result and the criterion for terminating a run**



SYMBOLIC REGRESSION #1 (WITH 21 FITNESS CASES)

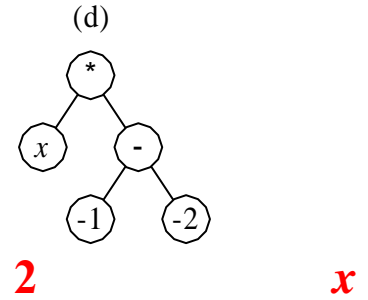
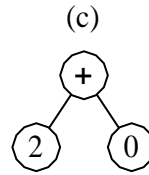
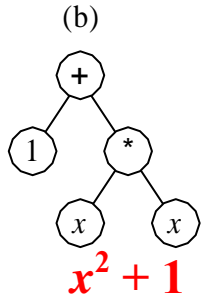
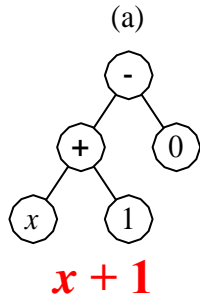
Independent variable X (Input)	Dependent Variable Y (Output)
-1.0	1.00
-0.9	0.91
-0.8	0.84
-0.7	0.79
-0.6	0.76
-0.5	0.75
-0.4	0.76
-0.3	0.79
-0.2	0.84
-0.1	0.91
0	1.00
0.1	1.11
0.2	1.24
0.3	1.39
0.4	1.56
0.5	1.75
0.6	1.96
0.7	2.19
0.8	2.44
0.9	2.71
1.0	3.00

TABLEAU—SYMBOLIC REGRESSION #1

	Objective:	Find a computer program with one input (independent variable x), whose output equals the values in the table in range from -1 to +1.
1	Terminal set:	$T = \{x, \text{constants}\}$
2	Function set:	$F = \{+, -, *, \%\}$ NOTE: The protected division function % returns a value of 1 when division by 0 is attempted (including 0 divided by 0)
3	Fitness:	The sum of the absolute value of the differences (errors), computed (in some way) over values of the independent variable x from -1.0 to +1.0, between the program's output and the target quadratic polynomial $x^2 + x + 1$.
4	Parameters:	Population size $M = 4$
5	Termination:	An individual emerges whose sum of absolute errors is less than 0.1

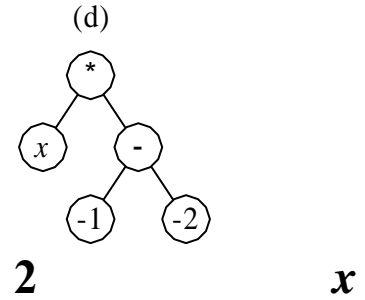
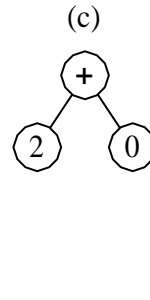
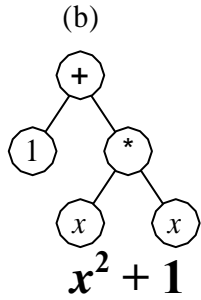
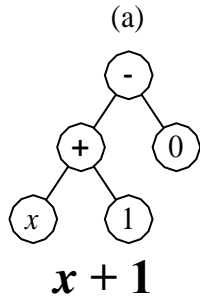
SYMBOLIC REGRESSION #1

INITIAL POPULATION OF FOUR INDIVIDUALS OF GENERATION 0

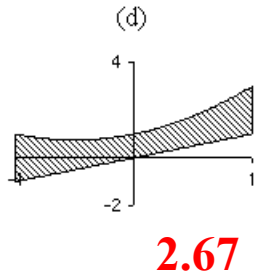
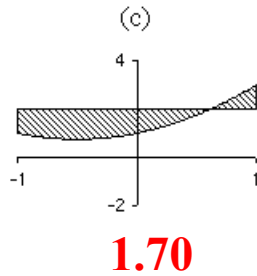
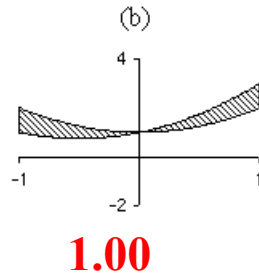
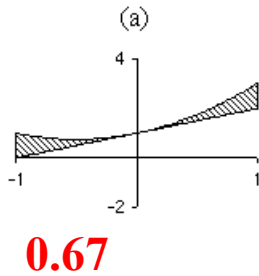


SYMBOLIC REGRESSION #1

INITIAL POPULATION OF FOUR INDIVIDUALS OF GENERATION 0

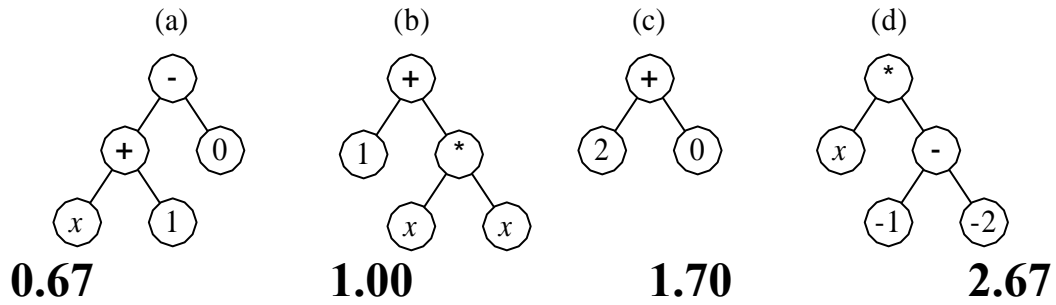


EVALUATE FITNESS

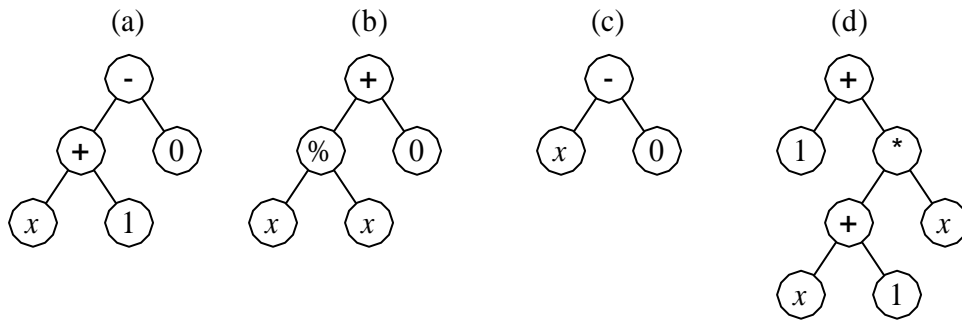


SYMBOLIC REGRESSION #1

GENERATION 0



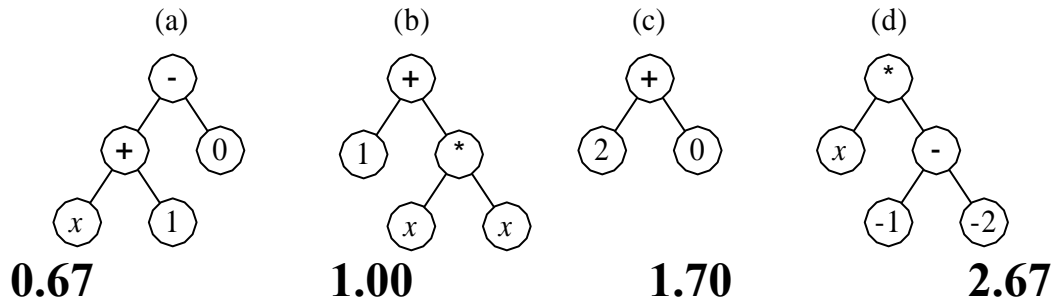
GENERATION 1



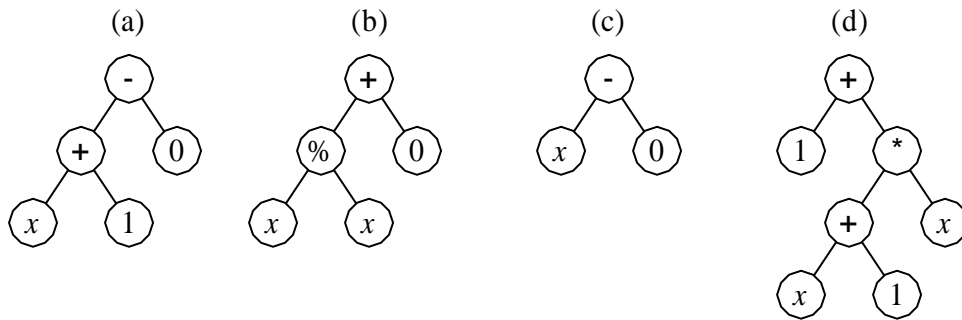
$x + 1$
Copy parent
(a)

SYMBOLIC REGRESSION #1

GENERATION 0



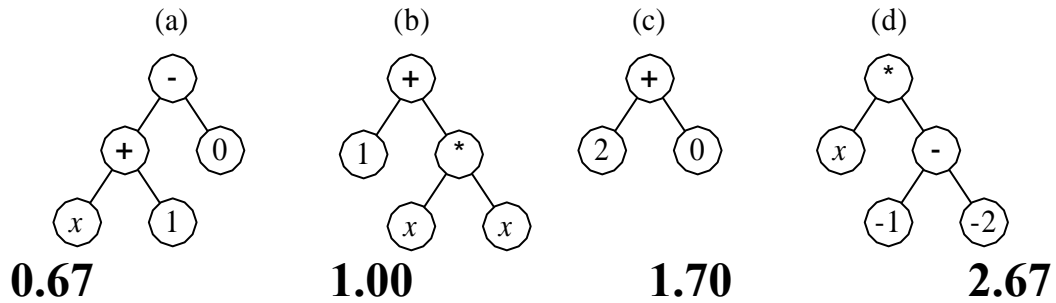
GENERATION 1



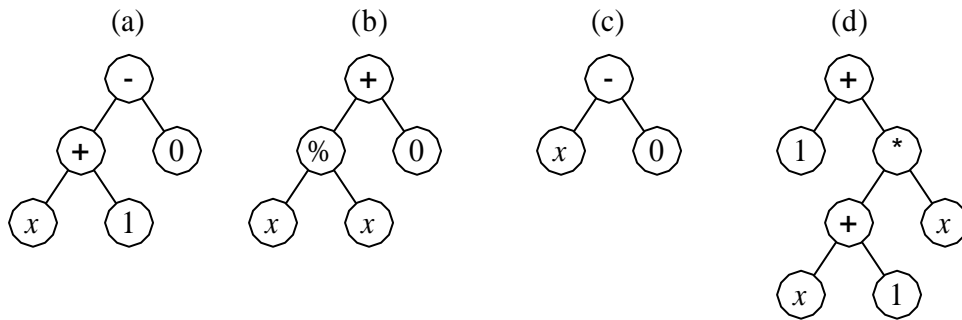
1	
Mutate parent (c)	
Picking "2" as mutation point	

SYMBOLIC REGRESSION #1

GENERATION 0



GENERATION 1



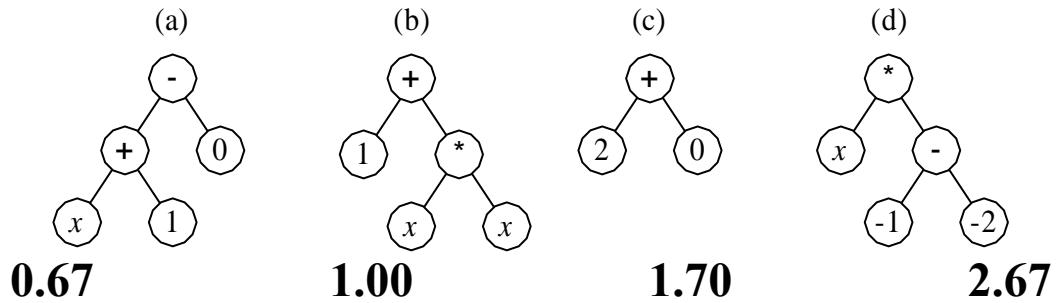
x

Crossover of (a) and (b)

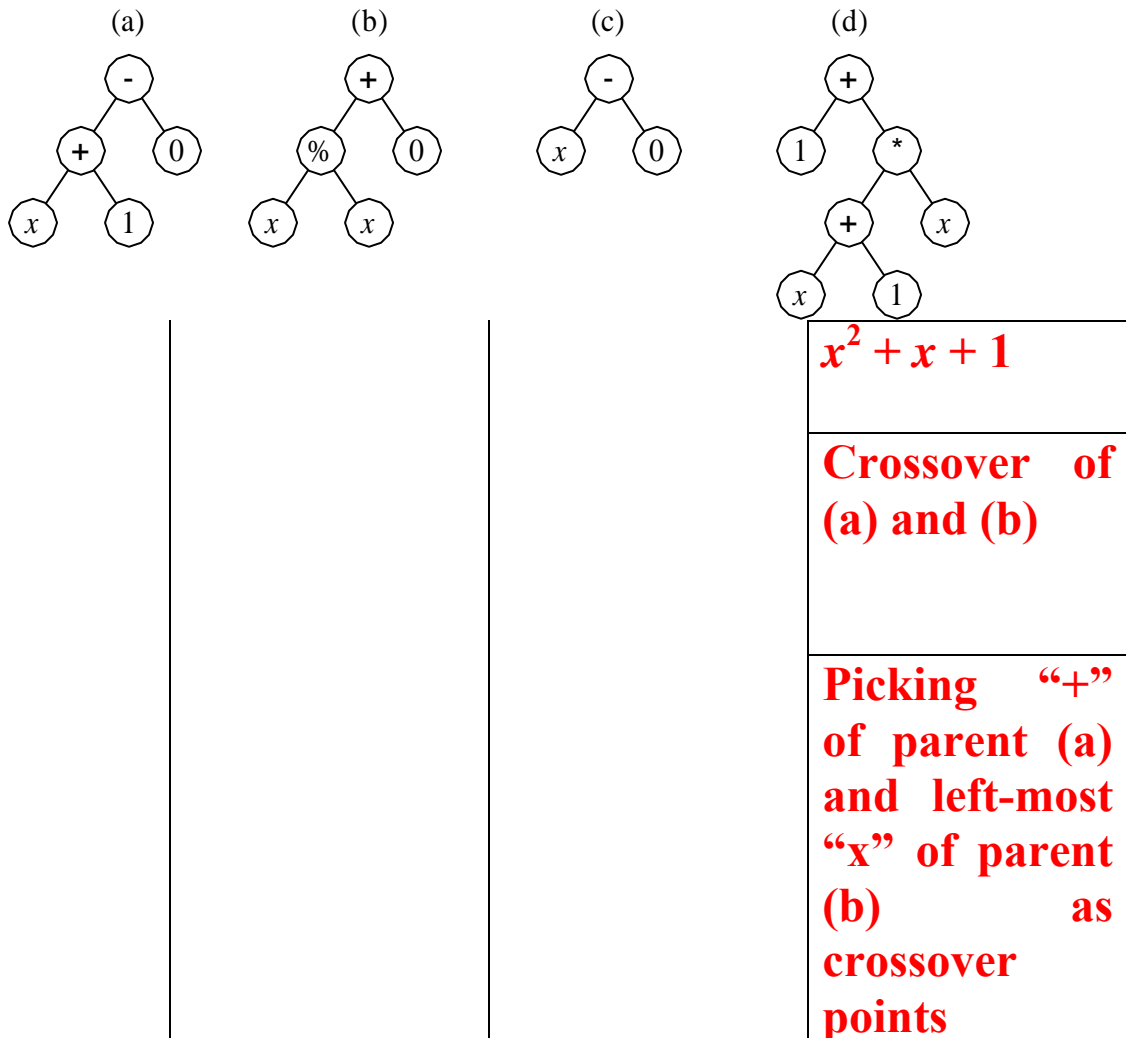
Picking “+” of parent (a) and left-most “x” of parent (b) as crossover points

SYMBOLIC REGRESSION #1

GENERATION 0

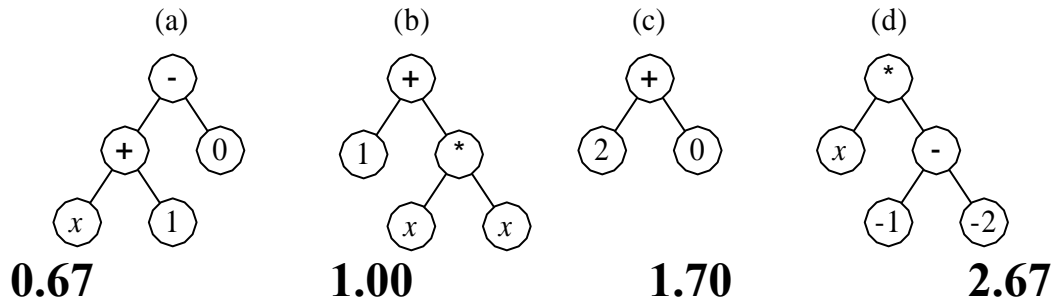


GENERATION 1

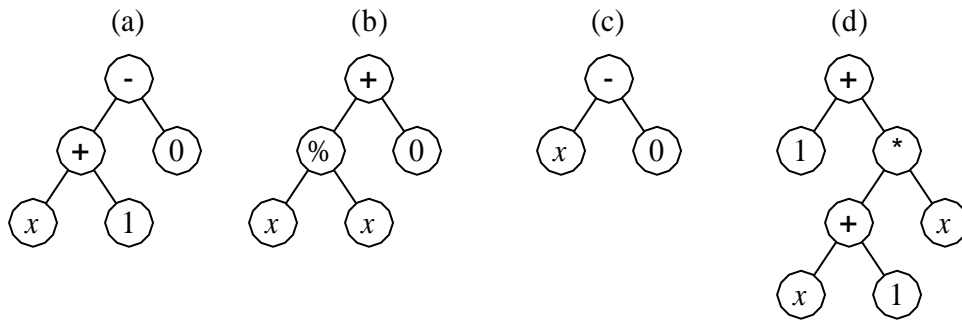


SYMBOLIC REGRESSION #1

GENERATION 0



GENERATION 1



$x + 1$	1	x	$x^2 + x + 1$
Copy parent (a)	Mutate parent (c)	Crossover of (a) and (b)	Crossover of (a) and (b)
	Picking “2” as mutation point	Picking “+” of parent (a) and left-most “x” of parent (b) as crossover points	Picking “+” of parent (a) and left-most “x” of parent (b) as crossover points

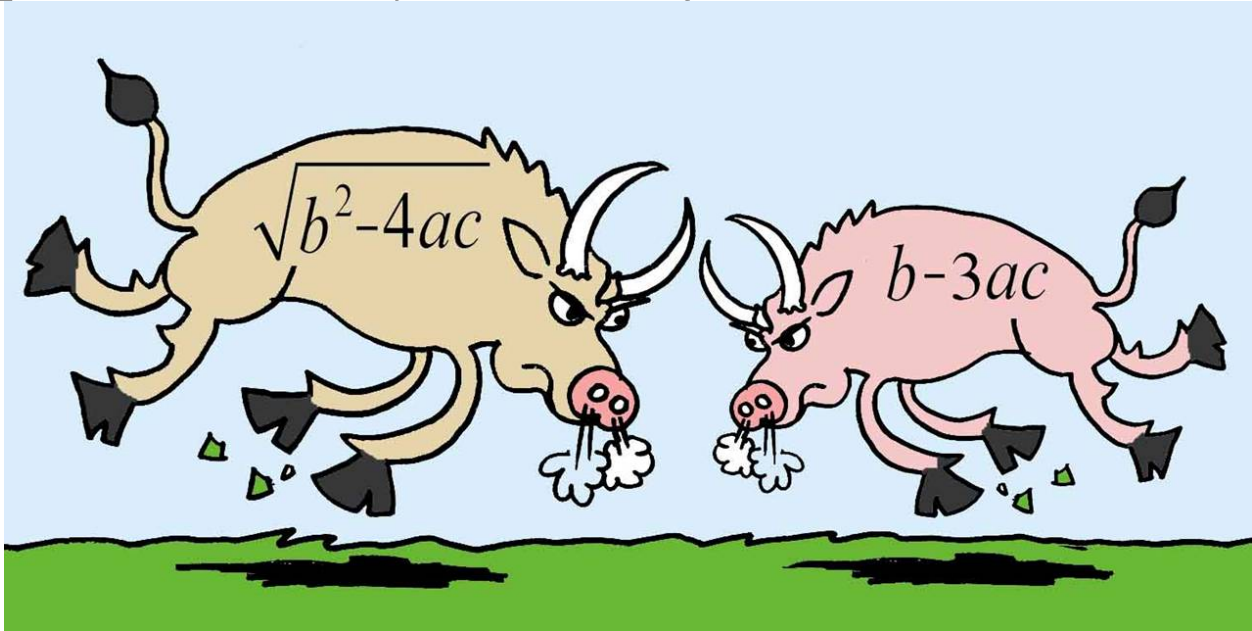
SYMBOLIC REGRESSION #1

OBSERVATIONS

- Genetic programming worked on this simple illustrative problem and produced quadratic polynomial $x^2 + x + 1$
- GP determined the size and shape of the solution
 - number of operations needed to solve the problem
 - size and shape of the program tree (topology)
 - content of the program tree (i.e., sequence of operations)
- The solution $x^2 + x + 1$ resulted from a recombination (crossover) of two “pretty good” elements, namely
 - the linear term x
 - the quadratic term $x^2 + 1$
- Cross validation is required. The answer is algebraically correct.

DARWINIAN NATURAL SELECTION

- All participants in the mutation, reproduction, and crossover operations are chosen from the current population *probabilistically based on fitness*



- 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

SYMBOLIC REGRESSION #2 (WITH 21 FITNESS CASES)

Independent variable (Input)	X	Dependent Variable (Output)	Y
-1.0		0.0000	
-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	

TABLEAU—SYMBOLIC REGRESSION #2

Objective:	Find a function of one independent variable, in symbolic form, that fits a given sample of 21 (x_i, y_i) data points
Terminal set:	x (the independent variable).
Function set:	$+$, $-$, $*$, $\%$, SIN , COS , EXP , RLOG
Fitness cases:	The given sample of 21 data points (x_i, y_i) where the x_i are in interval $[-1, +1]$.
Raw fitness:	The sum, taken over the 21 fitness cases, of the absolute value of difference between value of the dependent variable produced by the individual program and the target value y_i of the dependent variable.
Standardized fitness:	Equals raw fitness.
Hits:	Number of fitness cases (0–21) for which the value of the dependent variable produced by the individual program comes within 0.01 of the target value y_i of the dependent variable.
Wrapper:	None.

Parameters:	<ul style="list-style-type: none">• Population size, $M = 500$• Maximum number of generations to be run, $G = 51$• 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)
Success Predicate:	An individual program scores 21 hits.

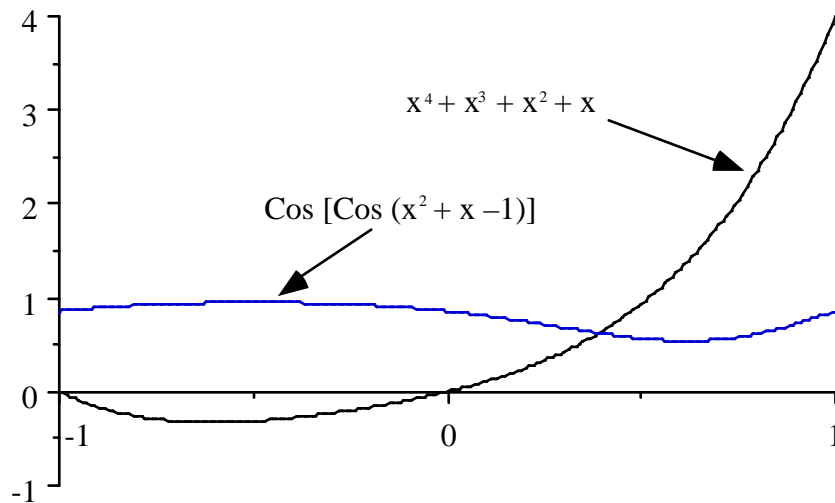
SYMBOLIC REGRESSION #2

**MEDIAN INDIVIDUAL IN GENERATION
0 WITH RAW FITNESS OF 23.67
(AVERAGE ERROR OF 1.3)**

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

Equivalent to

$$\text{Cos} [\text{Cos} (x^2 + x - 1)]$$



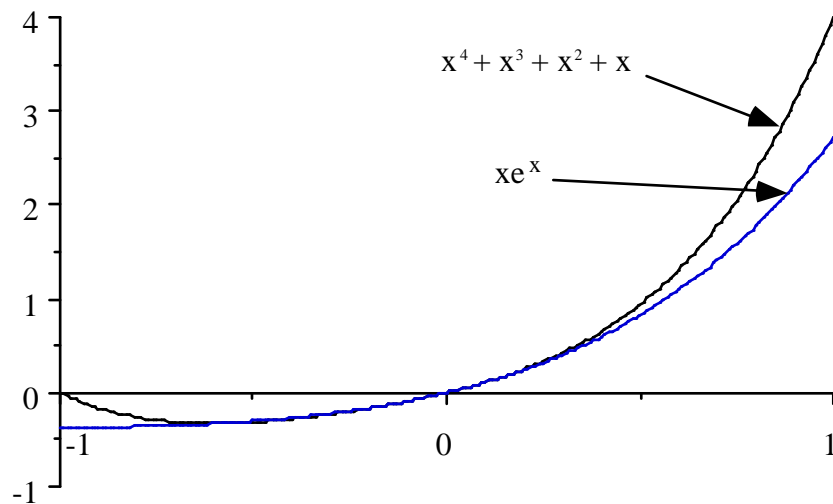
SYMBOLIC REGRESSION #2

**BEST-OF-GENERATION INDIVIDUAL IN
GENERATION 0 WITH RAW FITNESS OF
4.47 (AVERAGE ERROR OF 0.2)**

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

Equivalent to

$x e^x$



SYMBOLIC REGRESSION #2

**BEST-OF-GENERATION INDIVIDUAL IN
GENERATION 2 WITH RAW FITNESS OF
2.57 (AVERAGE ERROR OF 0.1)**

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

Equivalent to...

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

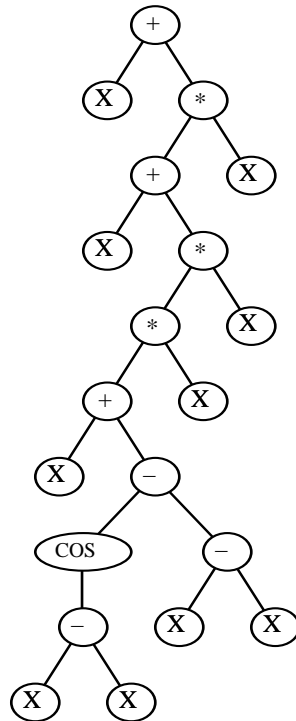
SYMBOLIC REGRESSION OF UNKNOWN FUNCTION #2

BEST-OF-RUN INDIVIDUAL IN GENERATION 34 WITH RAW FITNESS OF 0.00 (NO ERROR)

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

Equivalent to

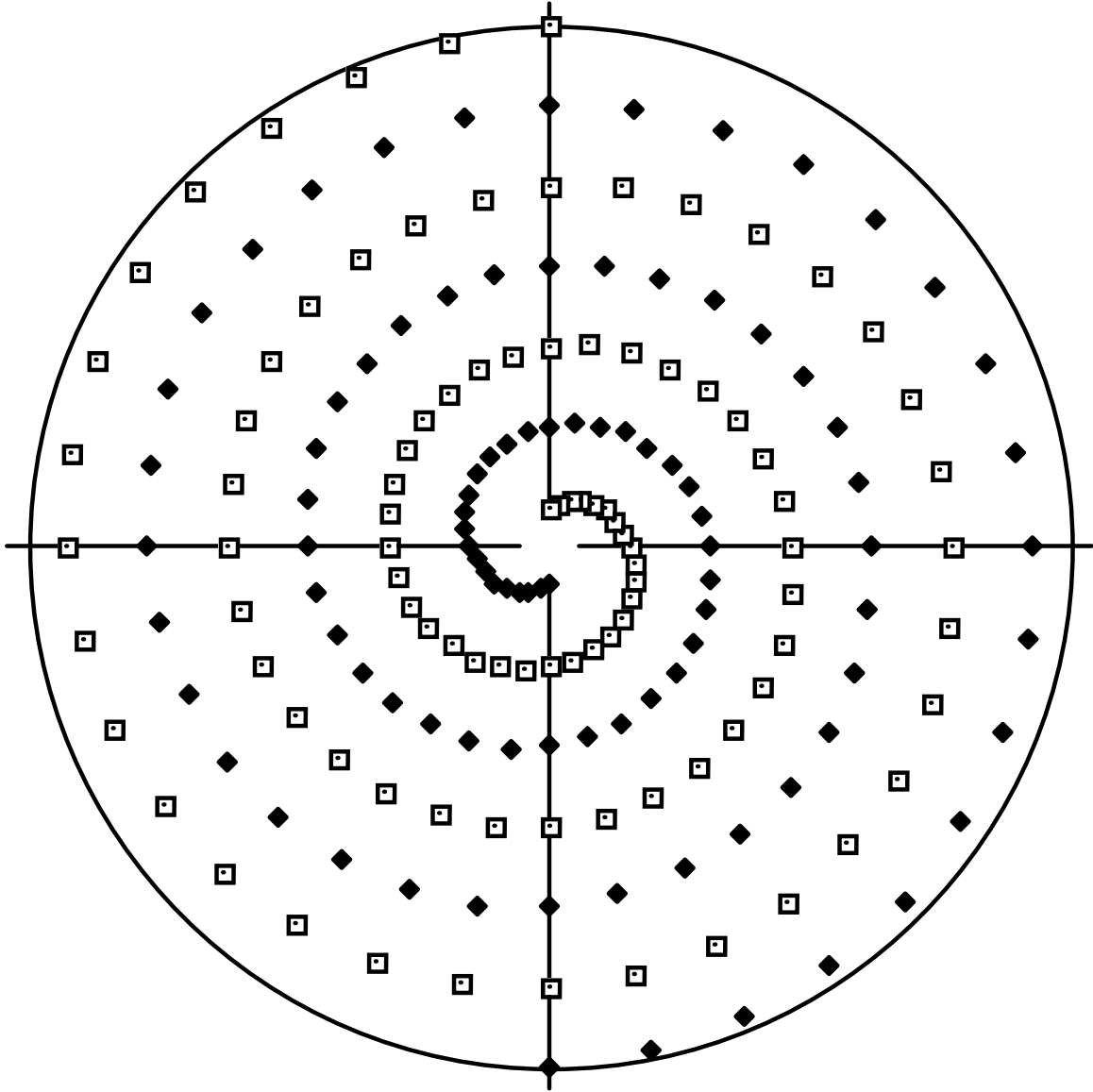
$$x^4 + x^3 + x^2 + x$$



SYMBOLIC REGRESSION OF FUNCTION #2—OBSERVATIONS

- The result is *not* how a human programmer would have done it
 - $\text{Cos}(X - X) = 1$
 - Not parsimonious
- The extraneous functions – SIN, EXP, RLOG, and RCOS are absent in the best individual of later generations because they are detrimental
 - $\text{Cos}(X - X) = 1$ is the exception that proves the rule
- GP operates the same whether the solution is linear, polynomial, a rational fraction of polynomials, exponential, trigonometric, etc.

CLASSIFICATION PROBLEM INTER-TWINED SPIRALS

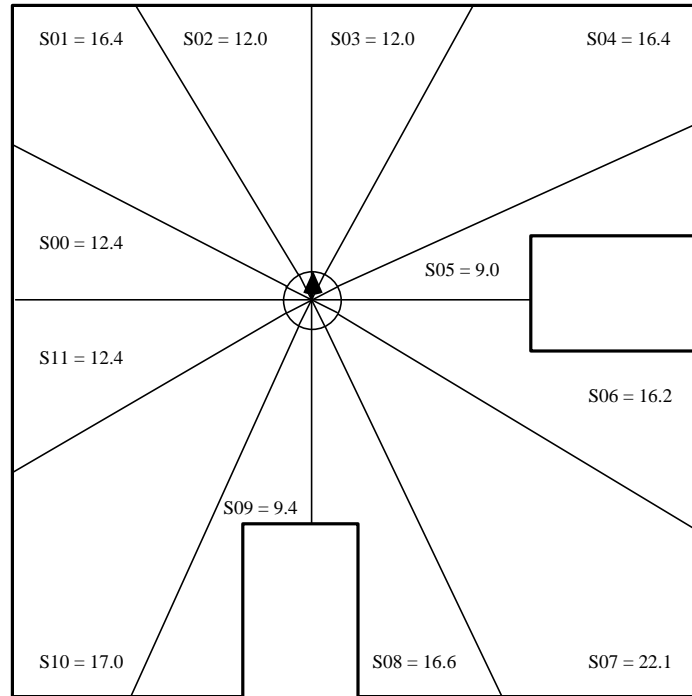


GP TABLEAU – INTERTWINED SPIRALS

Objective:	Find a program to classify a given point in the x - y plane to the red or blue spiral.
Terminal set:	x , y , \mathfrak{R} , where \mathfrak{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 correctly classified points (0 – 194)
Standardized fitness:	The maximum raw fitness (i.e., 194) minus the raw fitness.
Hits:	Equals raw fitness.
Wrapper:	Maps any individual program returning a positive value to class +1 (red) and maps all other values to class -1 (blue).
Parameters:	$M = 10,000$ (with over-selection). $G = 51$.
Success predicate:	An individual program scores 194 hits.

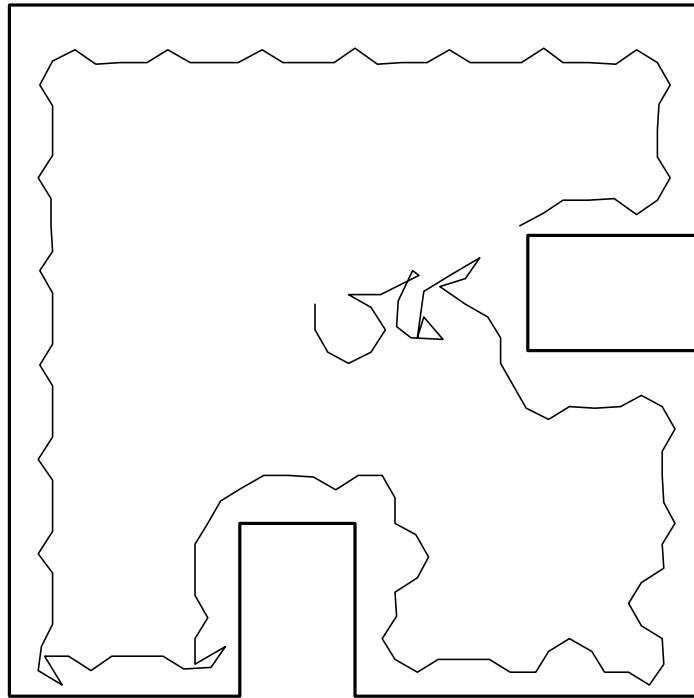
WALL-FOLLOWING PROBLEM

12 SONAR SENSORS

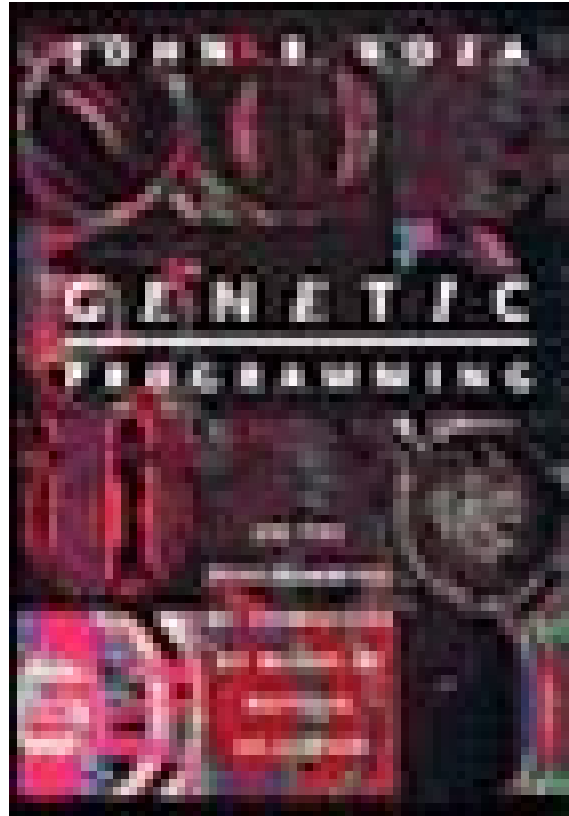


WALL-FOLLOWING PROBLEM BEST PROGRAM OF GENERATION 57

- Scores 56 hits (out of 56)
- 145point program tree



***GENETIC PROGRAMMING: ON THE
PROGRAMMING OF COMPUTERS BY
MEANS OF NATURAL SELECTION (MIT
PRESS, 1992)***



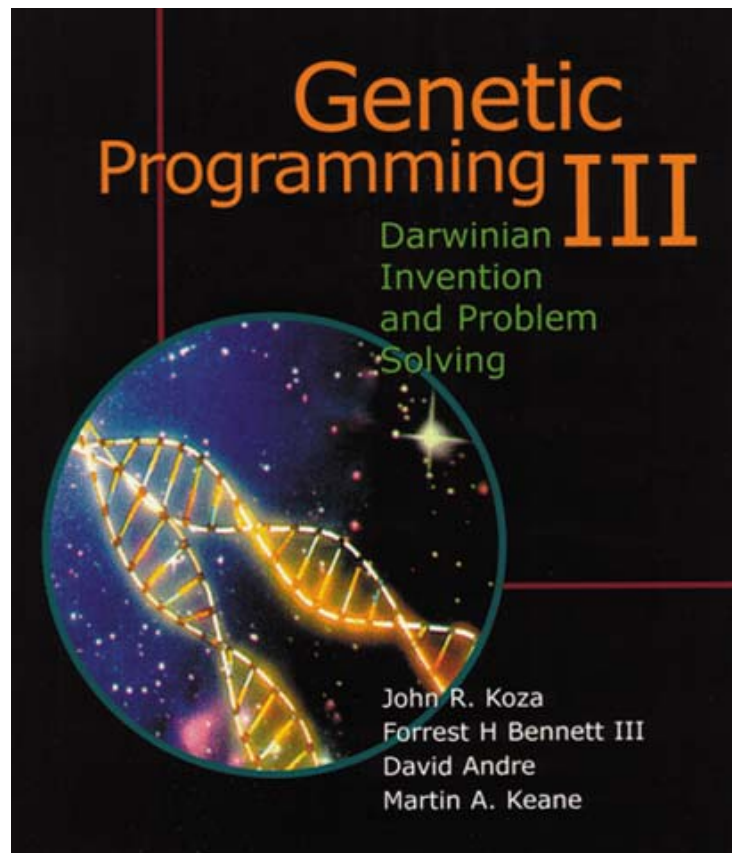
24 PROBLEMS SHOWN IN *GENETIC PROGRAMMING: THE MOVIE* (1992)

- **Symbolic Regression**
- **Intertwined Spirals**
- **Artificial Ant**
- **Truck Backer Upper**
- **Broom Balancing**
- **Wall Following**
- **Box Moving**
- **Discrete Pursuer-Evader Game**
- **Differential Pursuer-Evader Game**
- **Co-Evolution of Game-Playing Strategies**
- **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**

“DEVELOPMENTAL” GENETIC PROGRAMMING

- **We don't evolve the desired structure, but, instead, a set of instructions (i.e., a computer program) to construct the structure**

***GENETIC PROGRAMMING III:
DARWINIAN INVENTION AND PROBLEM
SOLVING***
**(KOZA, BENNETT, ANDRE, AND KEANE,
1999, MORGAN KAUFMANN)**



DEVELOPMENTAL GP

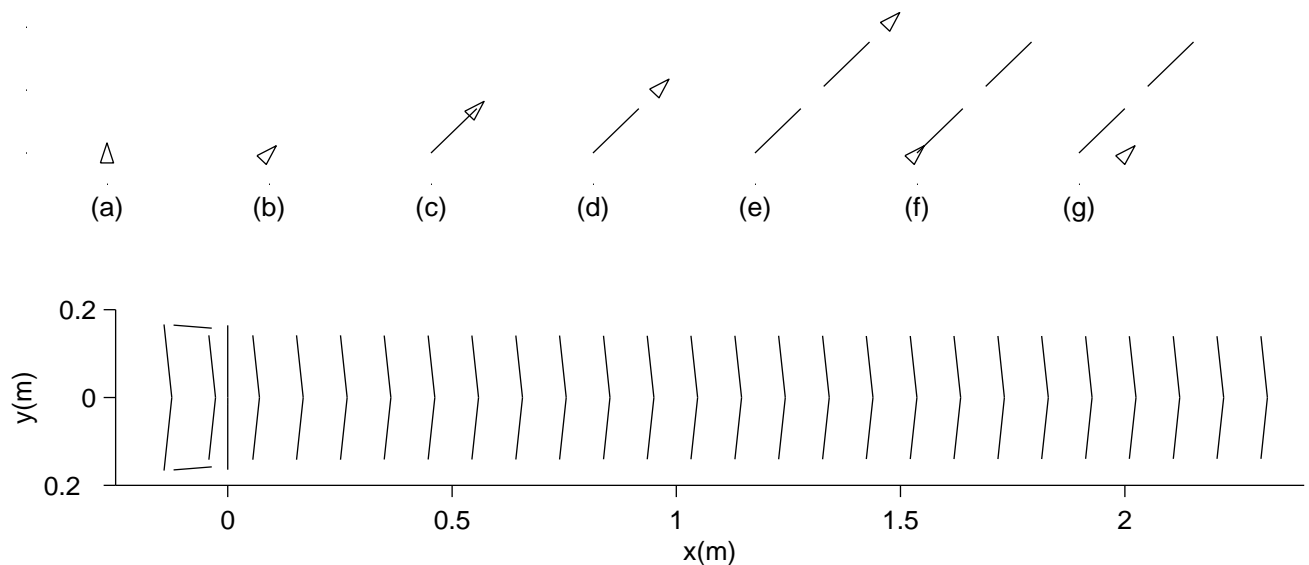
AUTOMATIC SYNTHESIS OF ANTENNA

EXAMPLE OF TURTLE FUNCTIONS

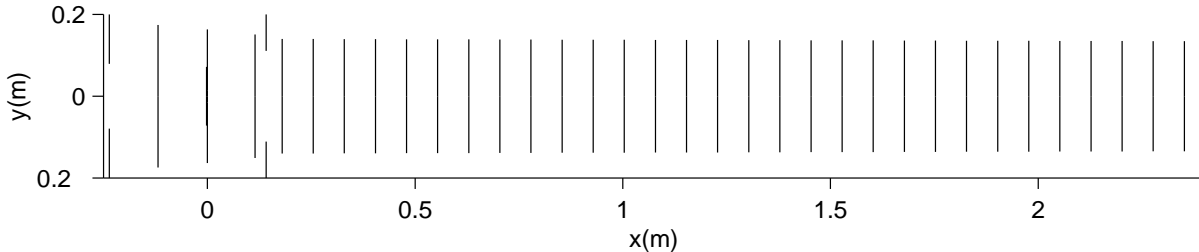
```

1 (PROGN3
2   (TURN-RIGHT 0.125)
3   (LANDMARK
4     (REPEAT 2
5       (PROGN2
6         (DRAW 1.0 HALF-MM-WIRE)
7         (DRAW 0.5 NO-WIRE) ) )
8   (TRANSLATE-RIGHT 0.125 0.75) ) )

```



BEST-OF-RUN ANTENNA FROM GENERATION 90

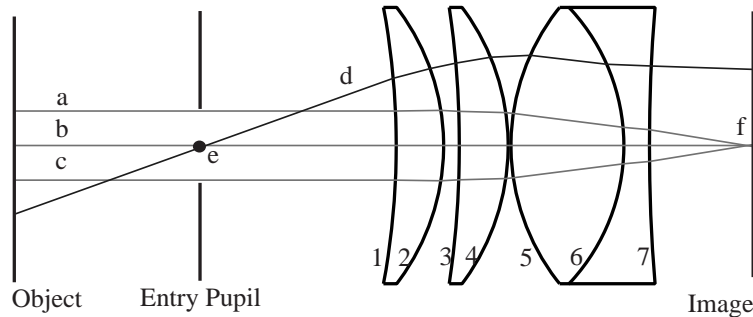


- **The GP run discovered**
 - (1) the number of reflectors (one),**
 - (2) the number of directors,**
 - (3) the fact that the driven element, the directors, and the reflector are all single straight wires,**
 - (4) the fact that the driven element, the directors, and the reflector are all arranged in parallel,**
 - (5) the fact that the energy source (via the transmission line) is connected only to single straight wire (the driven element) — that is, all the directors and reflectors are parasitically coupled**

- **Characteristics (3), (4), and (5) are essential characteristics of the Yagi-Uda antenna, namely an antenna with multiple parallel parasitically coupled straight-line directors, a single parallel parasitically coupled straight-line reflector, and a straight-line driven element.**

AUTOMATED DESIGN OF OPTICAL LENS SYSTEMS (KOZA, AL-SAKRAN, AND JONES 2005)

TACKABERRY-MULLER LENS SYSTEM

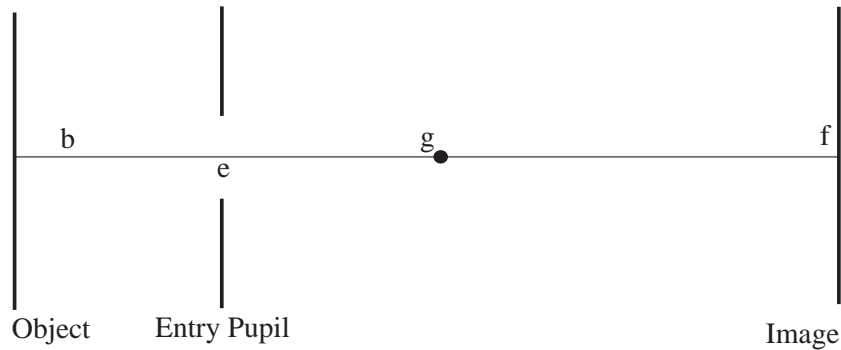


“PRESCRIPTION” (“LENS FILE”)

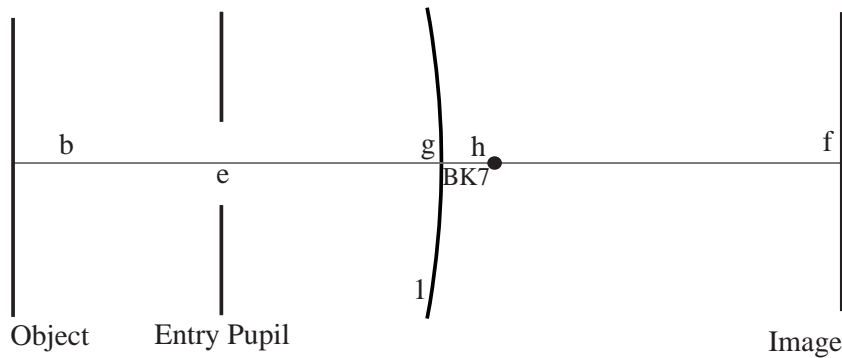
Surface	Distance	Radius	Material	Aperture
Object	10^{10}	flat	air	
Entry pupil	0.88	flat	air	0.18
1	0.21900	-3.5236	BK7	0.62
2	0.07280	-1.0527	air	0.62
3	0.22500	-4.4072	BK7	0.62
4	0.01360	-1.0704	air	0.62
5	0.52100	1.02491	BK7	0.62
6	0.11800	-0.9349	SF61	0.62
7	0.47485	7.94281	air	0.62
Image		flat		

DEVELOPMENTAL PROCESS

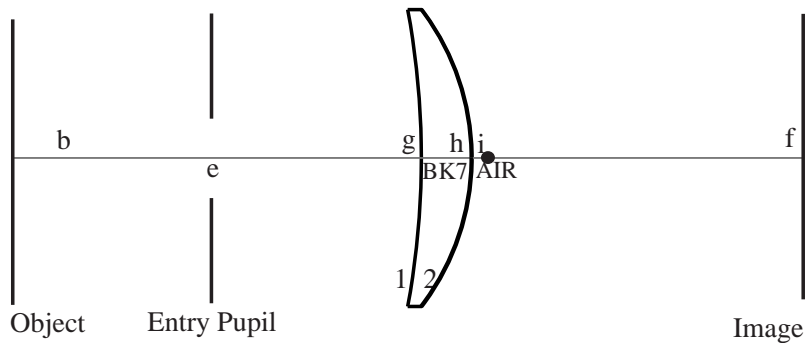
TURTLE STARTS AT POINT g ALONG MAIN AXIS b



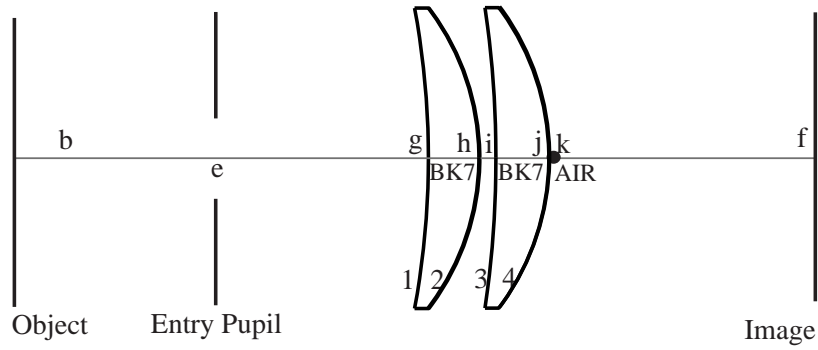
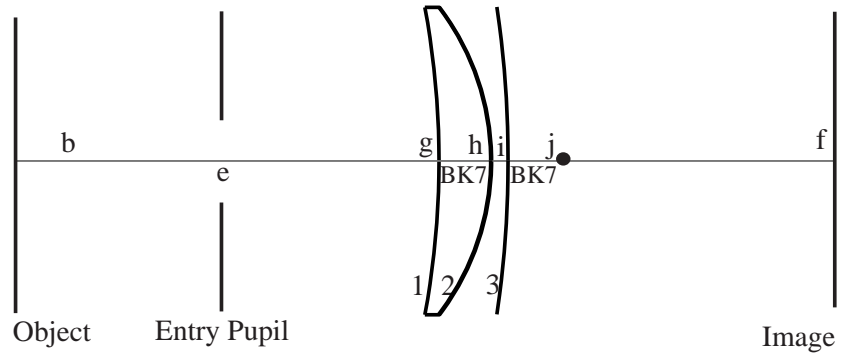
TURTLE INSERTS SURFACE 1



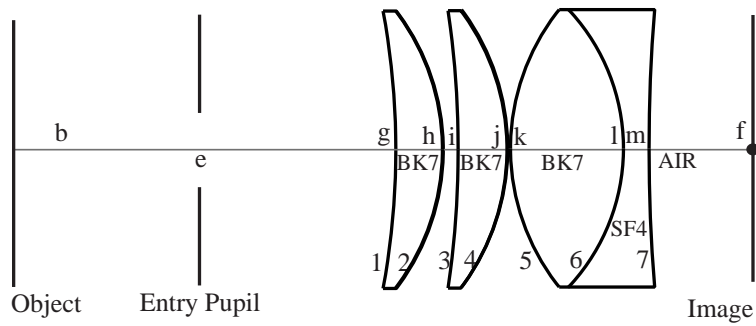
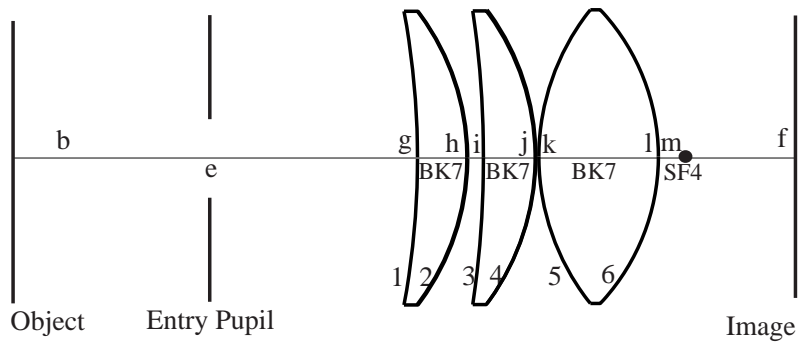
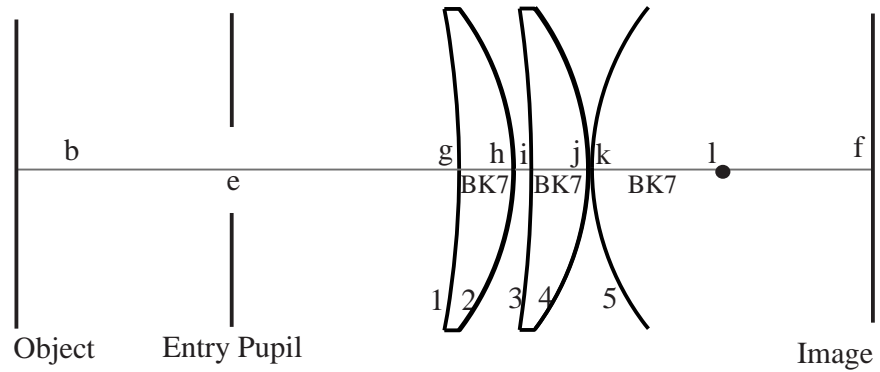
TURTLE INSERTS SURFACE 2



DEVELOPMENTAL PROCESS— CONTINUED



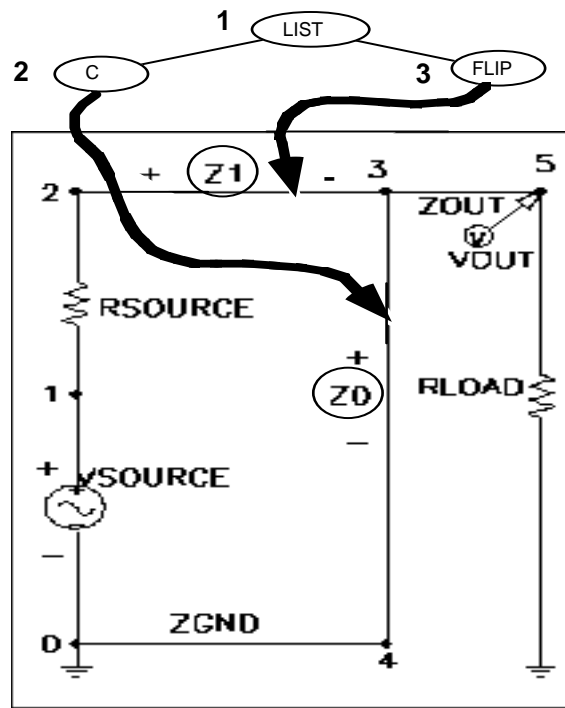
DEVELOPMENTAL PROCESS— CONTINUED



DEVELOPMENTAL GP

ANALOG ELECTRICAL CIRCUITS

THE INITIAL CIRCUIT



DEVELOPMENTAL GP

ANALOG ELECTRICAL CIRCUITS

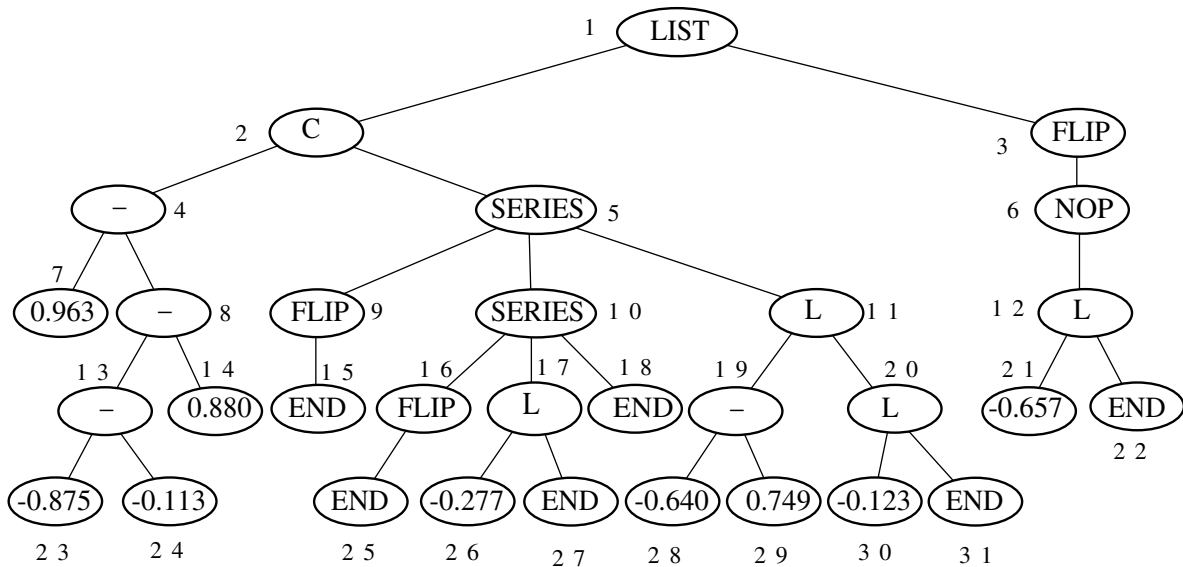
THE INITIAL CIRCUIT

- Initial circuit consists of embryo and test fixture
- Embryo has modifiable wires (e.g., Z0 AND Z1)
- Test fixture has input and output ports and usually has source resistor and load resistor. There are no modifiable wires (or modifiable components) in the test fixture.
- Circuit-constructing program trees consist of
 - Component-creating functions
 - Topology-modifying functions
 - Development-controlling functions
- Circuit-constructing program tree has one result-producing branch for each modifiable wire in embryo of the initial circuit

DEVELOPMENTAL GP

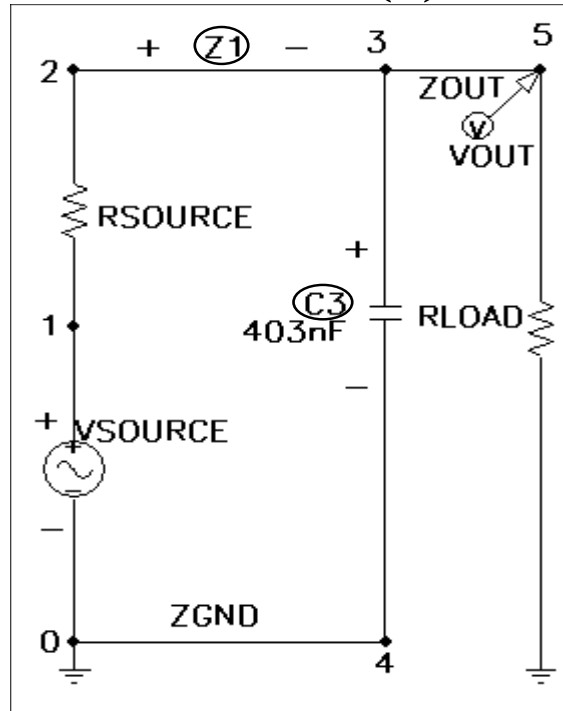
DEVELOPMENT OF A CIRCUIT FROM A CIRCUIT-CONSTRUCTING PROGRAM TREE AND THE INITIAL CIRCUIT

```
(LIST (C (- 0.963 (- (- -0.875
-0.113) 0.880)) (series (flip
end) (series (flip end) (L -
0.277 end) end) (L (- -0.640
0.749) (L -0.123 end)))) (flip
(nop (L -0.657 end))))
```



DEVELOPMENTAL GP

RESULT OF THE c (2) FUNCTION

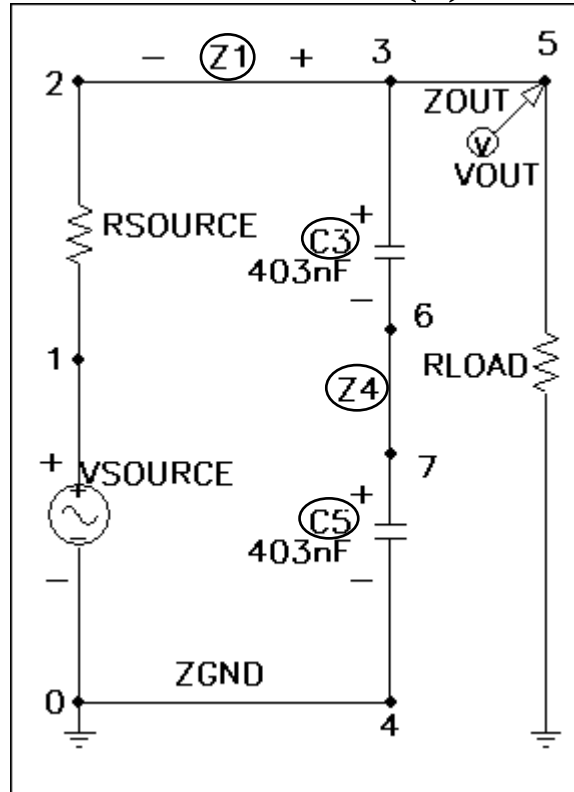


```
(LIST (C (- 0.963 (- (- -0.875
-0.113) 0.880)) (series (flip
end) (series (flip end) (L -
0.277 end) end) (L (- -0.640
0.749) (L -0.123 end)))) (flip
(nop (L -0.657 end))))
```

NOTE: Interpretation of arithmetic value

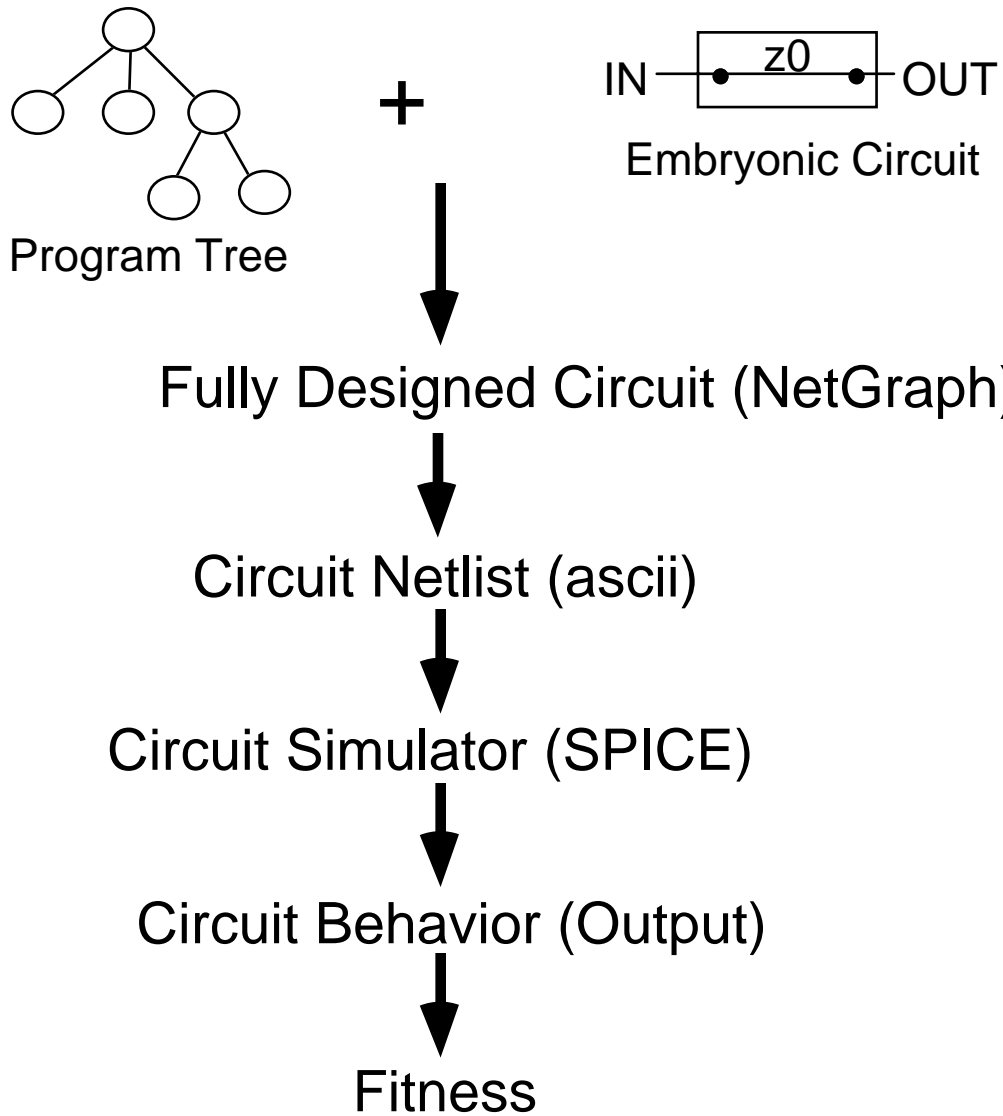
DEVELOPMENTAL GP

RESULT OF SERIES (5) FUNCTION

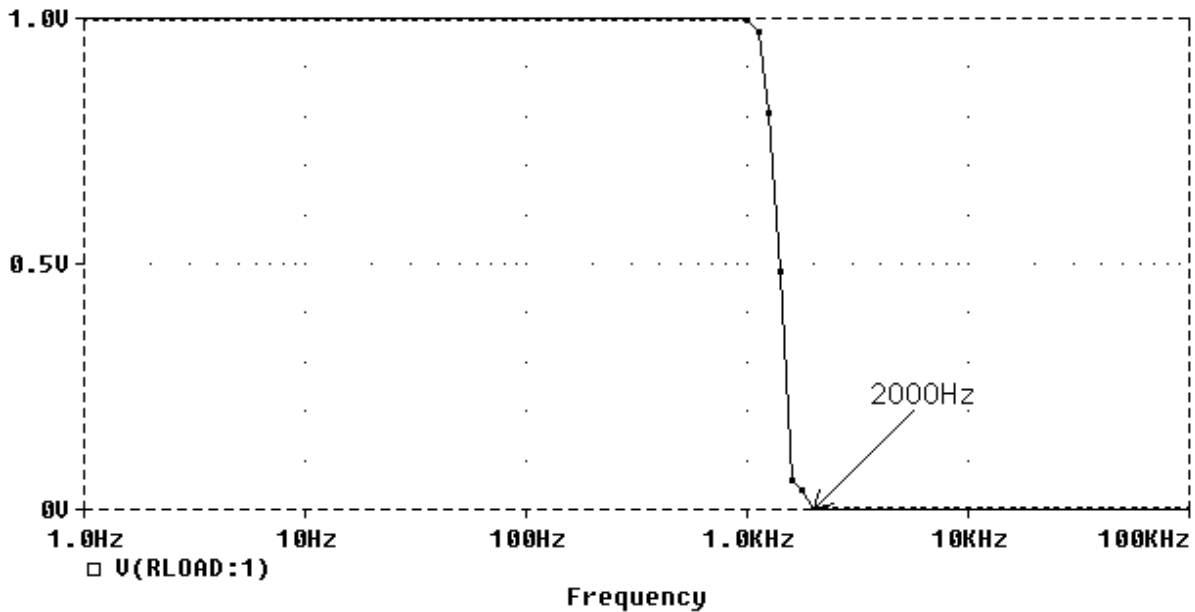


```
(LIST (C (- 0.963 (- (- -0.875
-0.113) 0.880)) (series (flip
end) (series (flip end) (L -
0.277 end) end) (L (- -0.640
0.749) (L -0.123 end)))) (flip
(nop (L -0.657 end))))
```

EVALUATION OF FITNESS OF A CIRCUIT



BEHAVIOR OF A LOWPASS FILTER VIEWED IN THE FREQUENCY DOMAIN



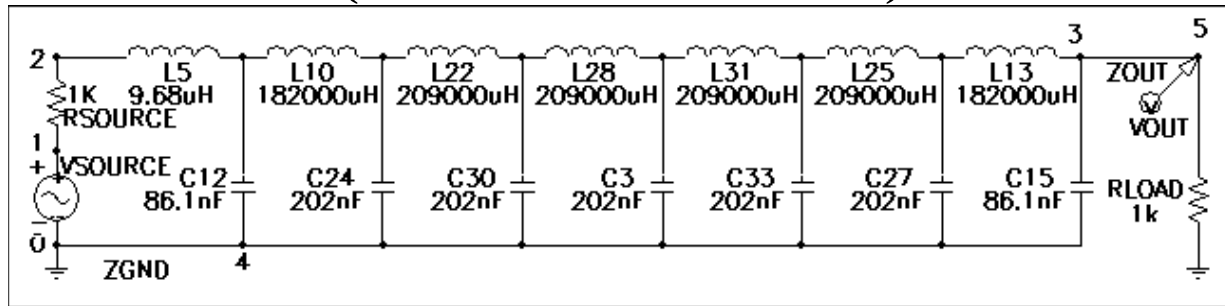
- Examine circuit's behavior for each of 101 frequency values chosen over five decades of frequency (from 1 Hz to 100,000 Hz) with each decade divided into 20 parts (using a logarithmic scale). The fitness measure
 - does not penalize ideal values
 - slightly penalizes acceptable deviations
 - heavily penalizes unacceptable deviations
- Fitness is $F(t) = \sum_{i=0}^{100} [W(f_i)d(f_i)]$
 - $f(i)$ is the frequency of fitness case i
 - $d(x)$ is the difference between the target and observed values at frequency of fitness case i
 - $W(y,x)$ is the weighting at frequency x

TABLEAU — LOWPASS FILTER

Objective:	Design a lowpass filter composed of inductors and capacitors with a passband below 1,000 Hz, a stopband above 2,000 Hz, a maximum allowable passband deviation of 30 millivolts, and a maximum allowable stopband deviation of 1 millivolt.
Test fixture and embryo:	One-input, one-output initial circuit with a source resistor, load resistor, and two modifiable wires.
Program architecture:	Two result-producing branches, RPB0 and RPB1 (i.e., one RPB per modifiable wire in the embryo).
Initial function set for the result-producing branches:	<p>For construction-continuing subtrees: $F_{\text{ccs-rpb-initial}} = \{C, L, \text{SERIES}, \text{PARALLEL0}, \text{FLIP}, \text{NOP}, \text{TWO_GROUND}, \text{TWO_VIA0}, \text{TWO_VIA1}, \text{TWO_VIA2}, \text{TWO_VIA3}, \text{TWO_VIA4}, \text{TWO_VIA5}, \text{TWO_VIA6}, \text{TWO_VIA7}\}.$</p> <p>For arithmetic-performing subtrees: $F_{\text{aps}} = \{+, -\}.$</p>
Initial terminal set for the result-producing branches:	<p>For construction-continuing subtrees: $T_{\text{ccs-rpb-initial}} = \{\text{END}\}.$</p> <p>For arithmetic-performing subtrees: $T_{\text{aps}} = \{\leftarrow\text{-smaller-reals}\}.$</p>
Fitness cases:	101 frequency values in an interval of five decades of frequency values between 1 Hz and 100,000 Hz.

Raw fitness:	Fitness is the sum, over the 101 sampled frequencies (fitness cases), of the absolute weighted deviation between the actual value of the output voltage that is produced by the circuit at the probe point and the target value for voltage. The weighting penalizes unacceptable output voltages much more heavily than deviating, but acceptable, voltages.
Standardized fitness:	Same as raw fitness.
Hits:	The number of hits is defined as the number of fitness cases (out of 101) for which the voltage is acceptable or ideal or that lie in the "don't care" band.
Wrapper:	None.
Parameters:	$M = 1,000$ to $320,000$. $G = 1,001$. $Q = 1,000$. $D = 64$. $B = 2\%$. $N_{rpb} = 2$. $S_{rpb} = 200$.
Result designation:	Best-so-far pace-setting individual.
Success predicate:	A program scores the maximum number (101) of hits.

EVOLVED CAMPBELL FILTER (7-RUNG LADDER)

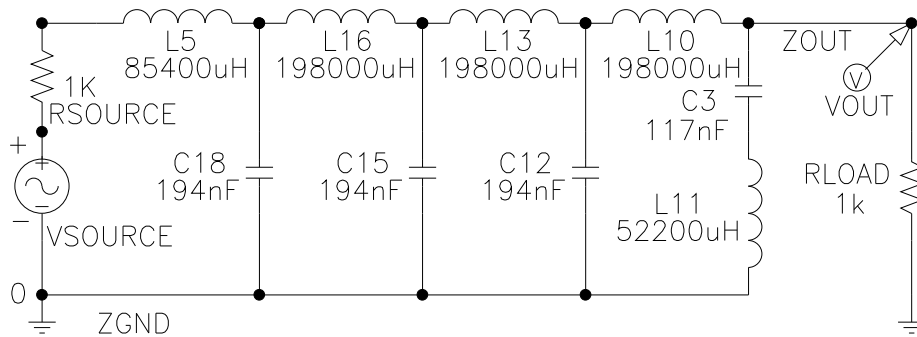


- This genetically evolved circuit infringes on U. S. patent 1,227,113 issued to George Campbell of American Telephone and Telegraph in 1917 (claim 2):

“An electric wave filter consisting of a connecting line of negligible attenuation composed of a plurality of sections, each section including a capacity element and an inductance element, one of said elements of each section being in series with the line and the other in shunt across the line, said capacity and inductance elements having precomputed values dependent upon the upper limiting frequency and the lower limiting frequency of a range of frequencies it is desired to transmit without attenuation, the values of said capacity and inductance elements being so proportioned that the structure transmits with practically negligible attenuation sinusoidal currents of all frequencies lying between said two limiting frequencies, while attenuating and approximately extinguishing currents of neighboring frequencies lying outside of said limiting frequencies.”

EVOLVED ZOBEL FILTER

- Infringes on U. S. patent 1,538,964 issued in 1925 to Otto Zobel of American Telephone and Telegraph Company for an “*M*-derived half section” used in conjunction with one or more “constant *K*” sections.
- One *M*-derived half section (C2 and L11)
- Cascade of three symmetric T-sections



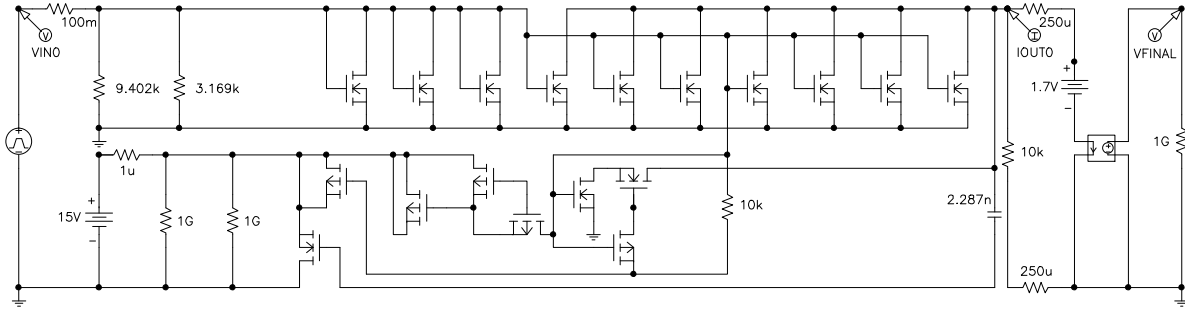
21 PREVIOUSLY PATENTED INVENTIONS REINVENTED BY GP

	Invention	Date	Inventor	Place	Patent
1	Darlington emitter-follower section	1953	Sidney Darlington	Bell Telephone Laboratories	2,663,806
2	Ladder filter	1917	George Campbell	American Telephone and Telegraph	1,227,113
3	Crossover filter	1925	Otto Julius Zobel	American Telephone and Telegraph	1,538,964
4	“M-derived half section” filter	1925	Otto Julius Zobel	American Telephone and Telegraph	1,538,964
5	Cauer (elliptic) topology for filters	1934–1936	Wilhelm Cauer	University of Gottingen	1,958,742, 1,989,545
6	Sorting network	1962	Daniel G. O’Connor and Raymond J. Nelson	General Precision, Inc.	3,029,413
7	Computational circuits	See text	See text	See text	See text
8	Electronic thermometer	See text	See text	See text	See text
9	Voltage reference circuit	See text	See text	See text	See text
10	60 dB and 96 dB amplifiers	See text	See text	See text	See text
11	Second-derivative controller	1942	Harry Jones	Brown Instrument Company	2,282,726
12	Philbrick circuit	1956	George Philbrick	George A. Philbrick Researches	2,730,679
13	NAND circuit	1971	David H. Chung and Bill H.	Texas Instruments Incorporated	3,560,760

			Terrell		
14	PID (proportional, integrative, and derivative) controller	1939	Albert Callender and Allan Stevenson	Imperial Chemical Limited	2,175,985
15	Negative feedback	1937	Harold S. Black	American Telephone and Telegraph	2,102,670, 2,102,671
16	Low-voltage balun circuit	2001	Sang Gug Lee	Information and Communications University	6,265,908
17	Mixed analog-digital variable capacitor circuit	2000	Turgut Sefket Aytur	Lucent Technologies Inc.	6,013,958
18	High-current load circuit	2001	Timothy Daun-Lindberg and Michael Miller	International Business Machines Corporation	6,211,726
19	Voltage-current conversion circuit	2000	Akira Ikeuchi and Naoshi Tokuda	Mitsumi Electric Co., Ltd.	6,166,529
20	Cubic function generator	2000	Stefano Cipriani and Anthony A. Takeshian	Conexant Systems, Inc.	6,160,427
21	Tunable integrated active filter	2001	Robert Irvine and Bernd Kolb	Infineon Technologies AG	6,225,859

POST-2000 PATENTED INVENTIONS

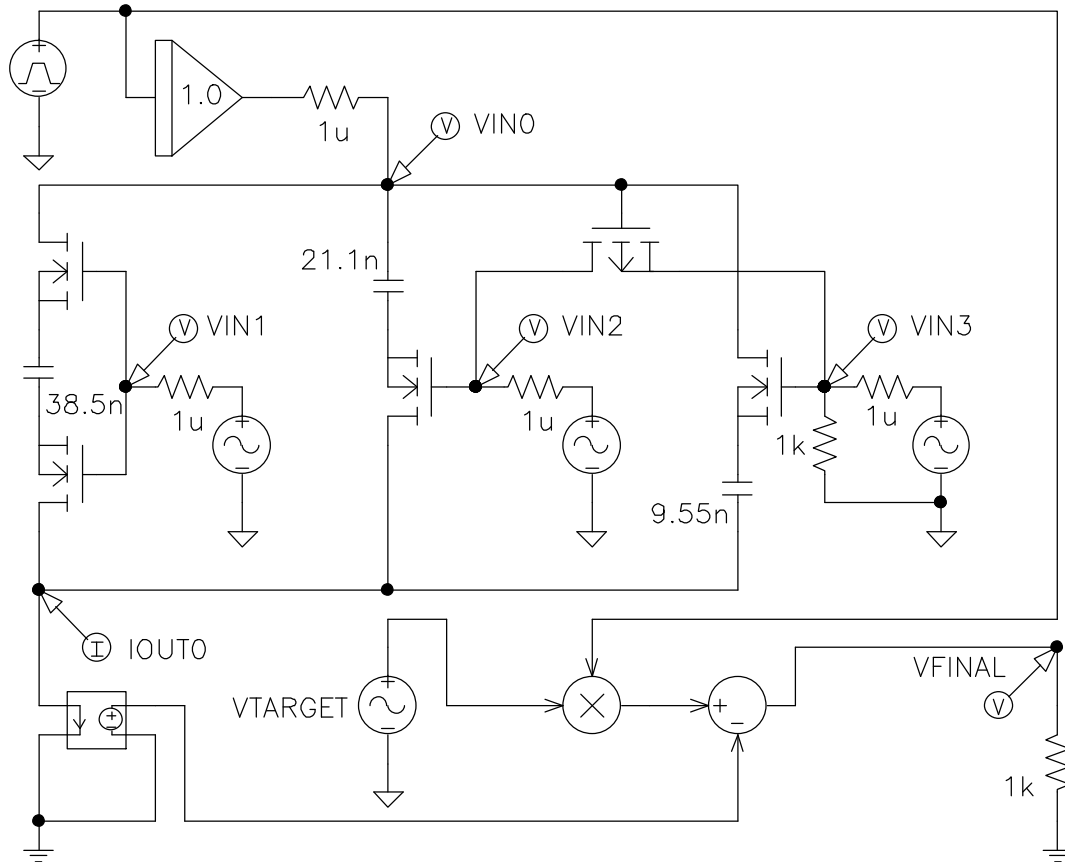
HIGH CURRENT LOAD CIRCUIT BEST-OF-RUN FROM GENERATION 114



POST-2000 PATENTED INVENTIONS

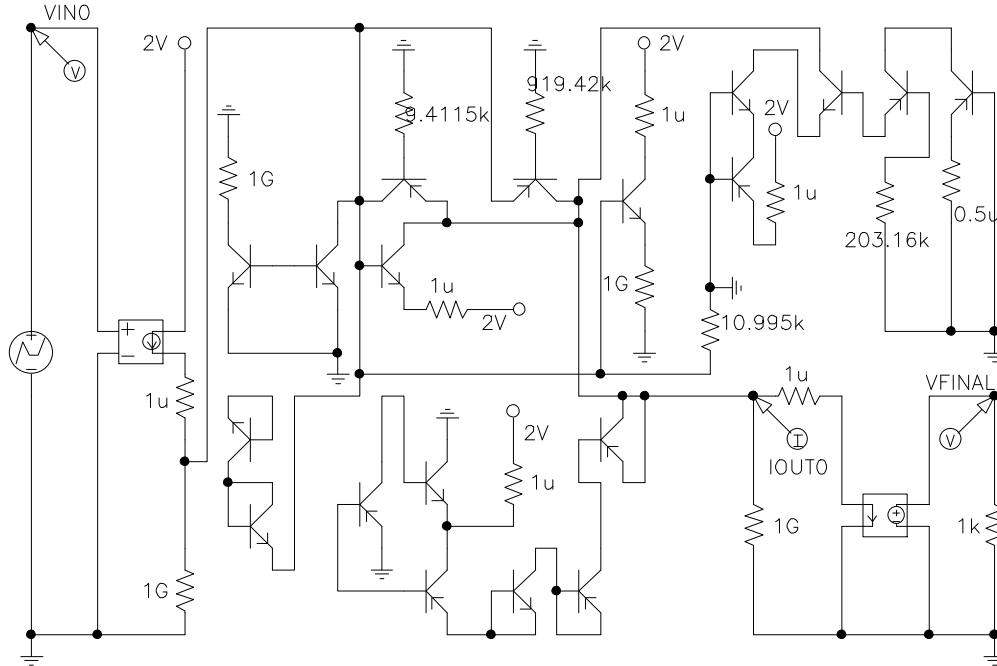
REGISTER-CONTROLLED CAPACITOR CIRCUIT

SMALLEST COMPLIANT FROM GENERATION 98



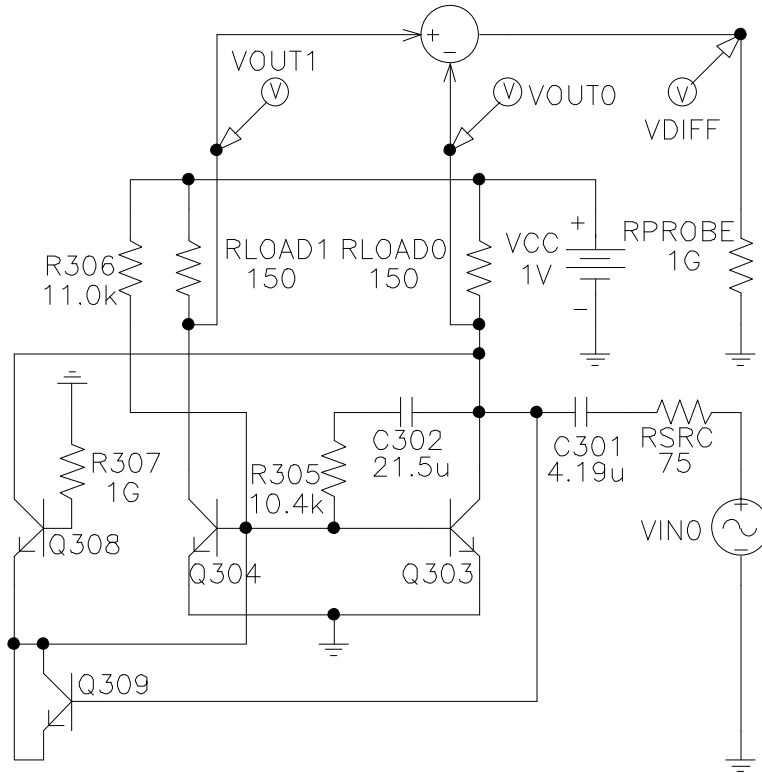
POST-2000 PATENTED INVENTIONS

LOW-VOLTAGE CUBIC SIGNAL GENERATION CIRCUIT BEST-OF-RUN FROM GENERATION 182



POST-2000 PATENTED INVENTIONS

LOW-VOLTAGE BALUN CIRCUIT BEST EVOLVED FROM GENERATION 84

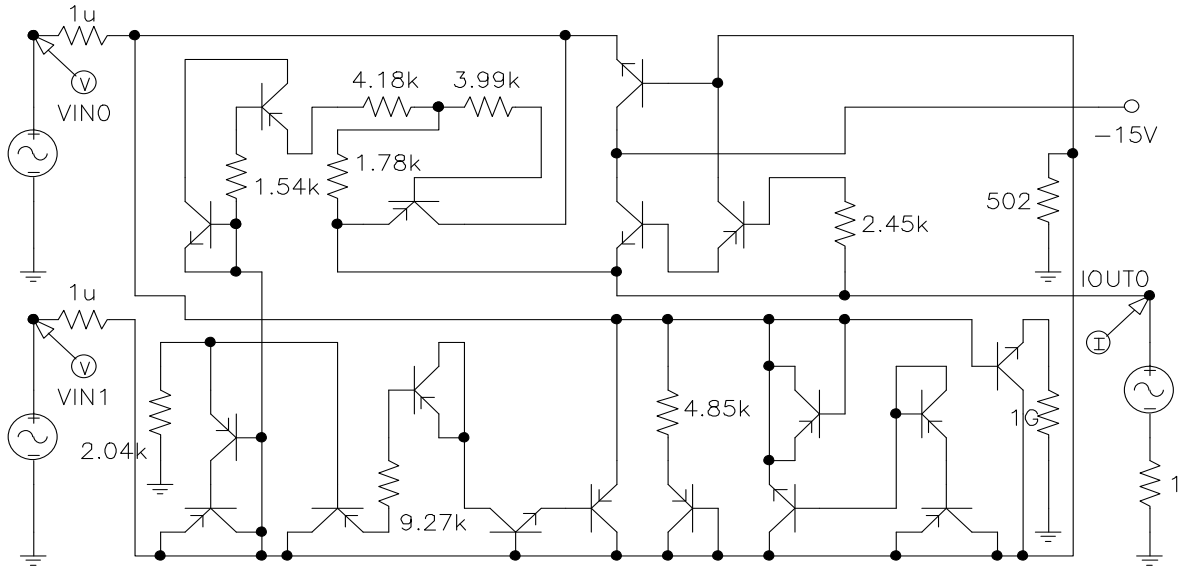


POST-2000 PATENTED INVENTIONS

VOLTAGE-CURRENT-CONVERSION

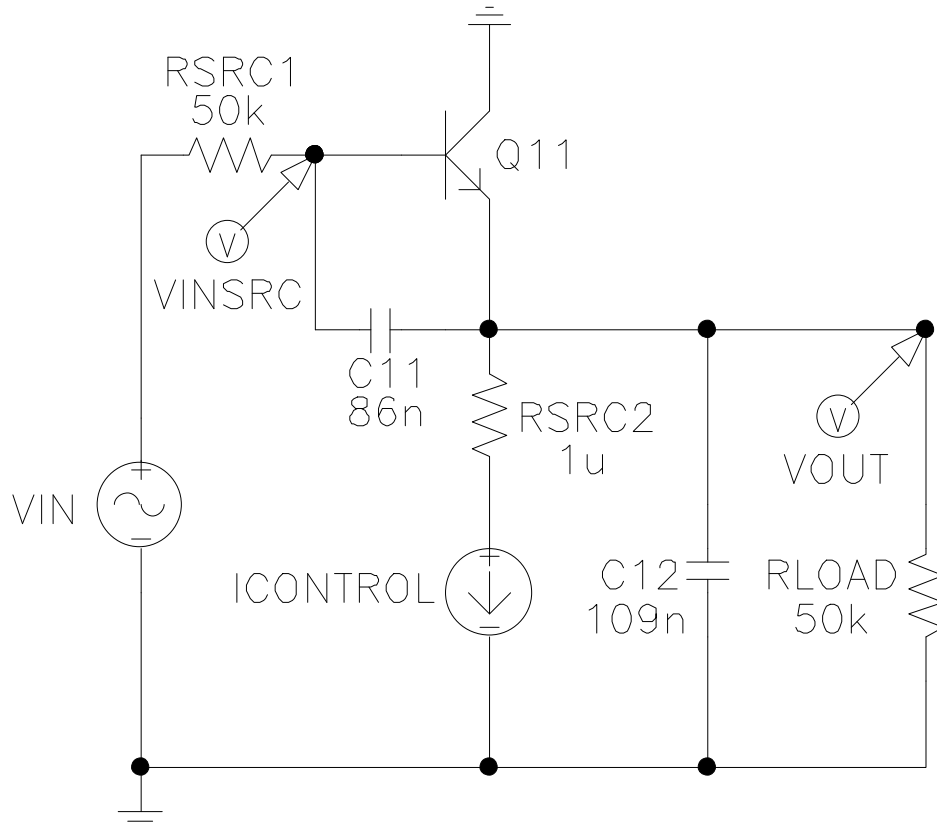
CIRCUIT

BEST-OF-RUN FROM GENERATION 109



POST-2000 PATENTED INVENTIONS

TUNABLE INTEGRATED ACTIVE FILTER — GENERATION 50



2 PATENTED INVENTIONS CREATED BY GENETIC PROGRAMMING

Keane, Martin A., Koza, John R., and Streeter, Matthew J.
2005. *Apparatus for Improved General-Purpose PID and
Non-PID Controllers*. U. S. Patent 6,847,851. Filed July
12, 2002. Issued January 25, 2005

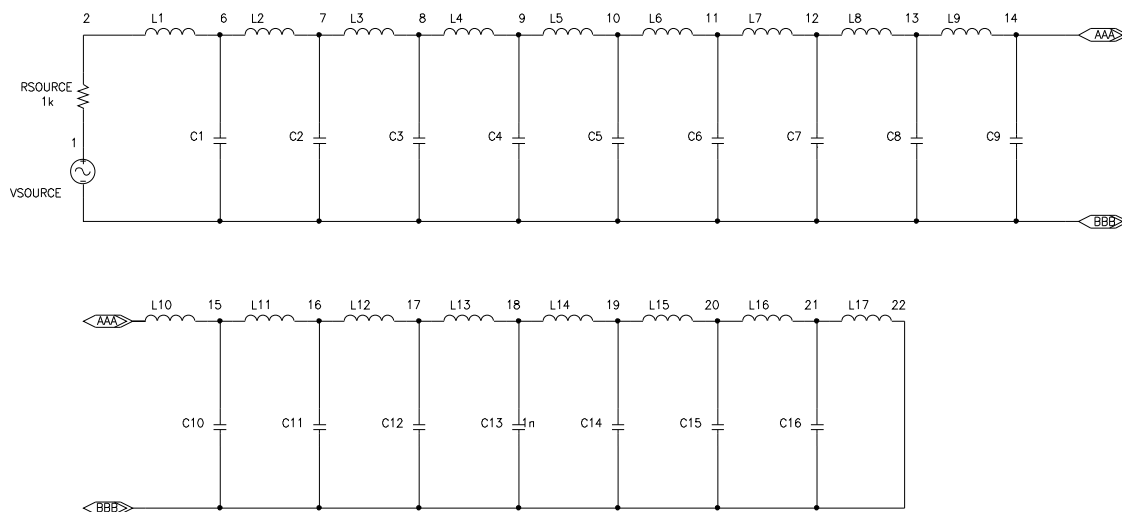


NOVELTY-DRIVEN EVOLUTION

EXAMPLE OF LOWPASS FILTER

- Two factors in fitness measure
 - Circuit's behavior in the frequency domain
 - Largest number of nodes and edges (circuit components) of a subgraph of the given circuit that is isomorphic to a subgraph of a template representing the prior art. Graph isomorphism algorithm with the cost function being based on the number of shared nodes and edges (instead of just the number of nodes).

PRIOR ART TEMPLATE

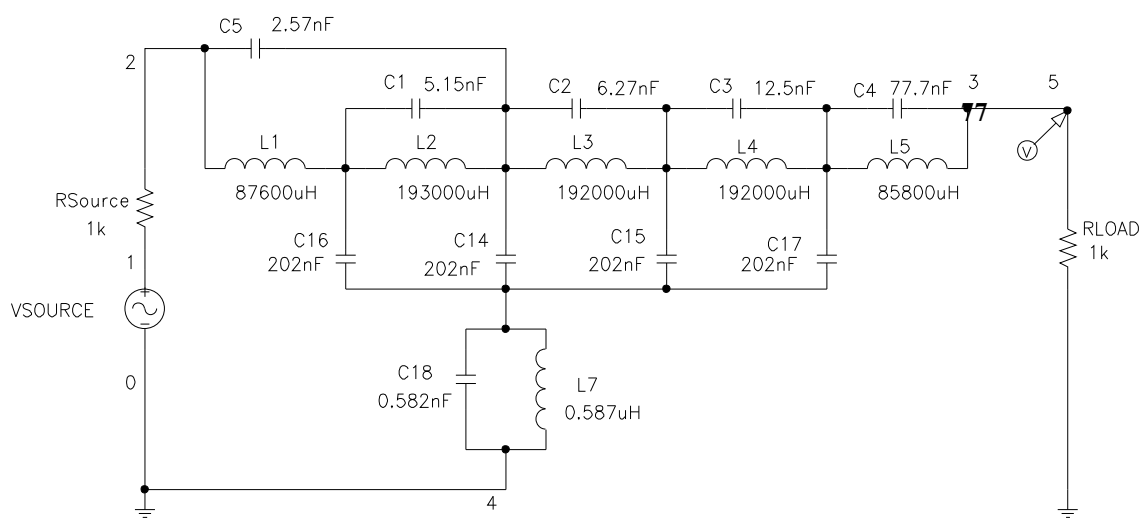


NOVELTY-DRIVEN EVOLUTION — CONTINUED

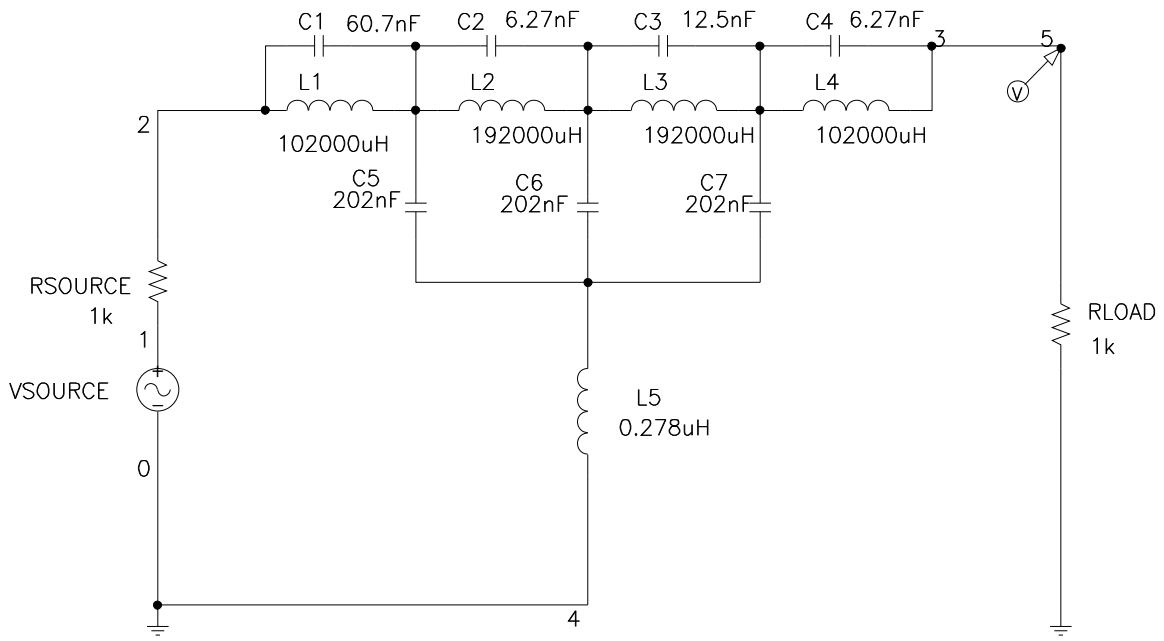
- For circuits not scoring the maximum number (101) of hits, the fitness of a circuit is the product of the two factors.
- For circuits scoring 101 hits (100%-compliant individuals), fitness is the number of shared nodes and edges divided by 10,000.

FITNESS OF EIGHT 100%-COMPLIANT CIRCUITS

Solution	Frequency factor	Isomorphism factor	Fitness
1	0.051039	7	0.357273
2	0.117093	7	0.819651
3	0.103064	7	0.721448
4	0.161101	7	1.127707
5	0.044382	13	0.044382
6	0.133877	7	0.937139
7	0.059993	5	0.299965
8	0.062345	11	0.685795



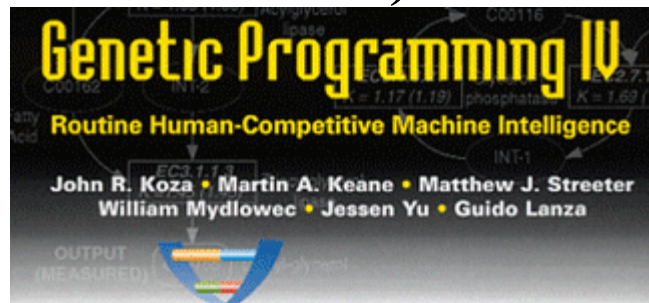
SOLUTION NO. 1



SOLUTION NO. 5

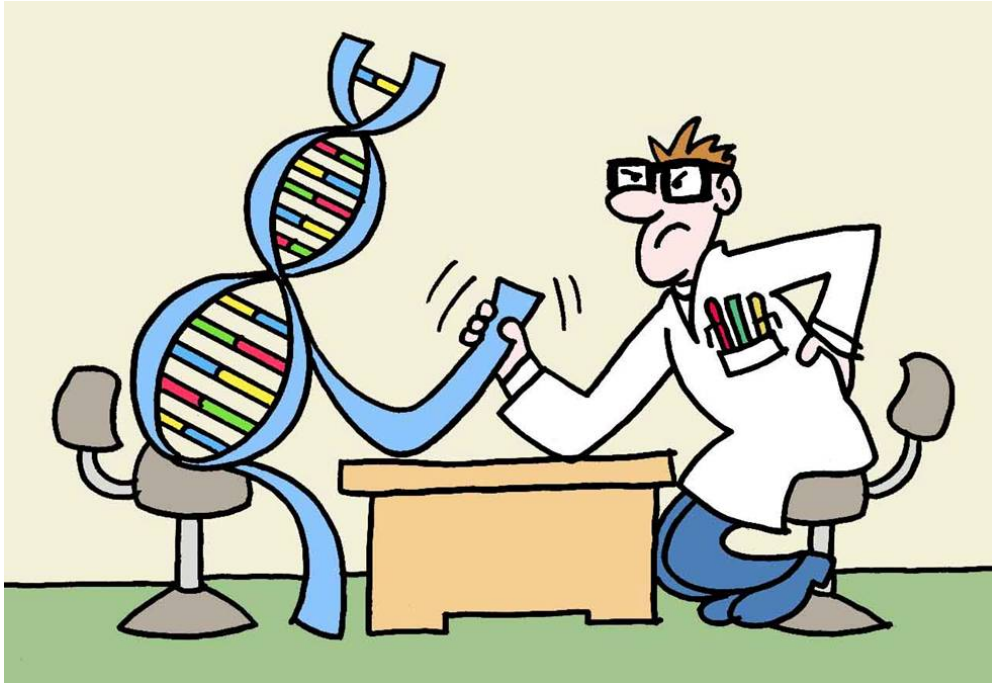
***GENETIC PROGRAMMING IV. ROUTINE
HUMAN-COMPETITIVE MACHINE
INTELLIGENCE***

**(KOZA, KEANE, STREETER,
MYDLOWEC, YU, AND LANZA,
KLUWER ACADEMIC PUBLISHERS,
2003)**



Kluwer Academic Publishers

EIGHT CRITERIA FOR HUMAN- COMPETITIVENESS



CRITERIA FOR “HUMAN-COMPETITIVENESS”

- **The result is equal or better than human-designed solution to the same problem**
- **The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.**
- **The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.**
- **The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.**
- **The result is publishable in its own right as a new scientific result independent of the fact that the result was mechanically created.**
- **The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.**
- **The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.**

CRITERIA FOR “HUMAN- COMPETITIVENESS”

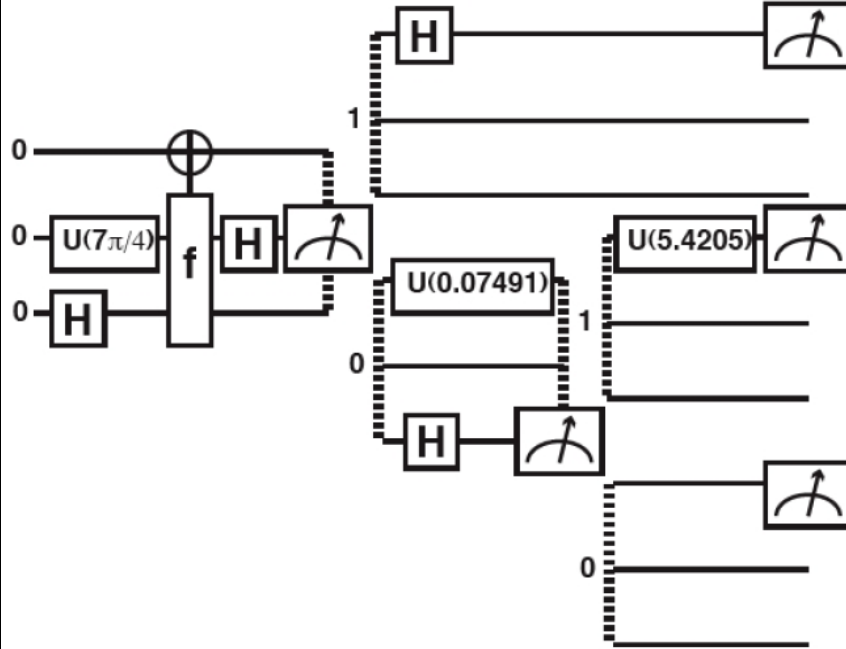
- **The result solves a problem of indisputable difficulty in its field.**
- **The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).**

HUMAN-COMPETITIVE RESULTS PRODUCED BY GP

Claimed instance	Picture
<p>Creation of a better-than-classical quantum algorithm for the Deutsch-Jozsa “early promise” problem</p> <p>Spector, Barnum, and Bernstein 1998</p>	
<p>Creation of a better-than-classical quantum algorithm for Grover’s database search problem</p> <p>Spector, Barnum, and Bernstein 1999</p>	

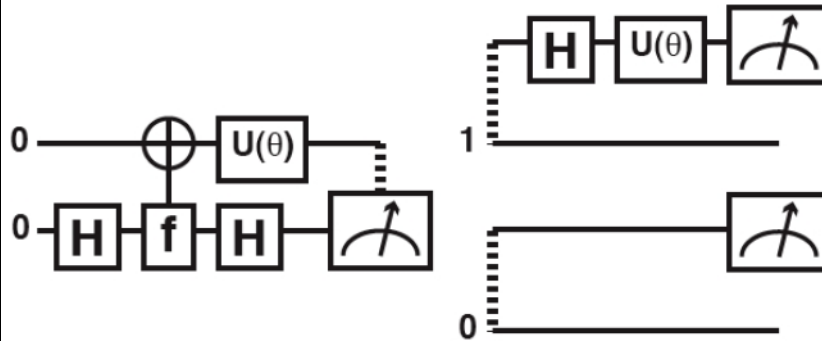
Creation of a quantum algorithm for the depth-two AND/OR query problem that is better than any previously published result

Spector, Barnum, Bernstein, and Swamy 1999; Barnum, Bernstein, and Spector 2000



Creation of a quantum algorithm for the depth-one OR query problem that is better than any previously published result

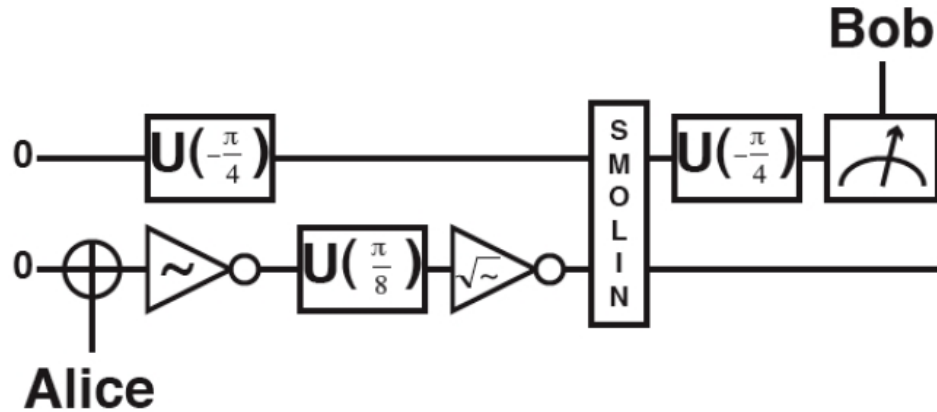
Barnum, Bernstein, and Spector 2000



$$\theta = 5.96143477$$

Creation of a protocol for communicating information through a quantum gate that was previously thought not to permit such communication

Spector and Bernstein 2003



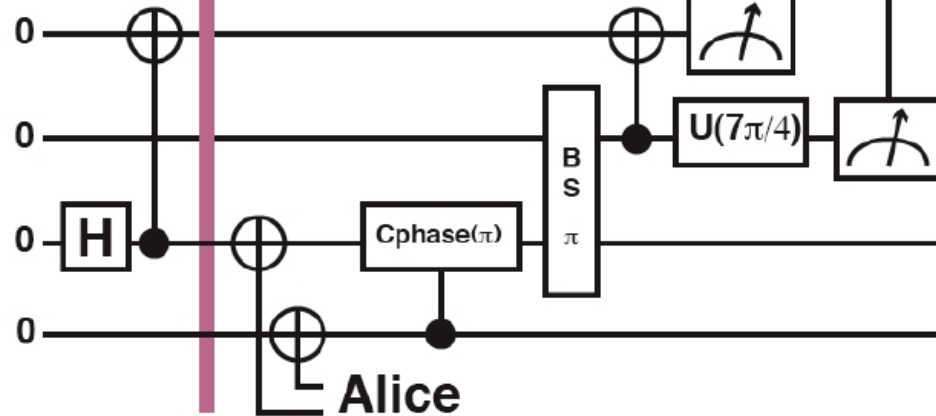
To understand one needs to know what the Smolin gate is and this is given in smolin-gate.jpg

$$\text{Smolin} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Creation of a novel variant of quantum dense coding

Spector and Bernstein 2003

Entangle

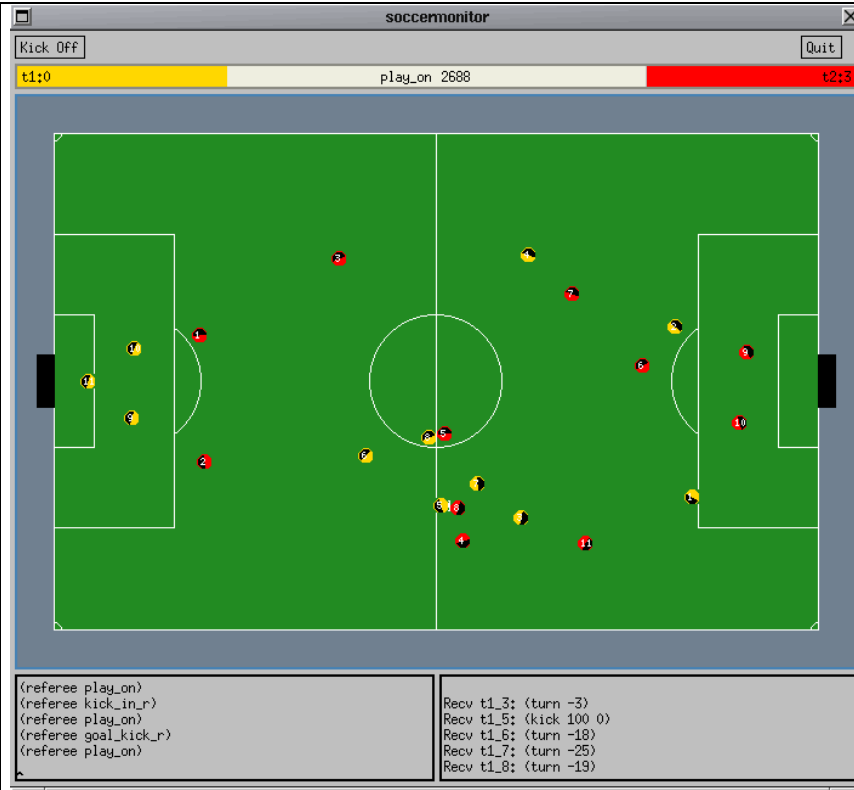


To understand one needs to know what the BS gate is and this is given to bs-gate.jpg

$$BS(\theta) = \begin{bmatrix} \cos(\theta) & 0 & 0 & \sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & 0 & -\cos(\theta) \end{bmatrix}$$

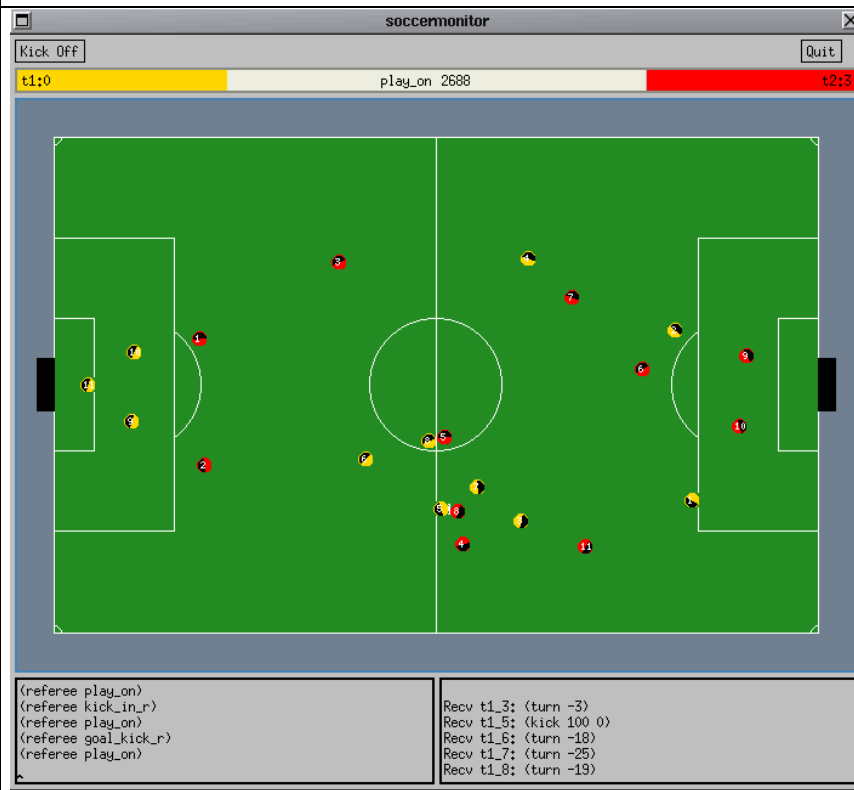
Creation of a soccer-playing program that won its first two games in the Robo Cup 1997 competition

Luke 1998



Creation of a soccer-playing program that ranked in the middle of the field of 34 human-written programs in the Robo Cup 1998 competition

Andre and Teller 1999



Creation of four different algorithms for the transmembrane segment identification problem for proteins

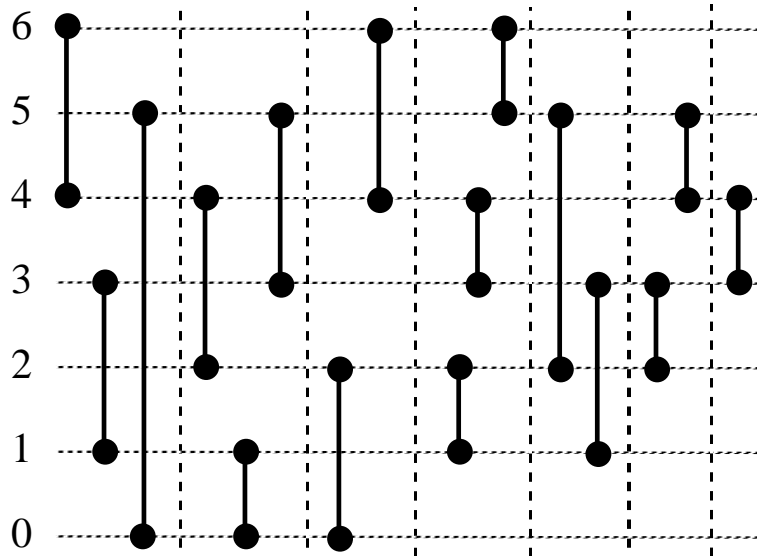
Sections 18.8 and 18.10 of *Genetic Programming II* and sections 16.5 and 17.2 of *Genetic Programming III*

"0-2-4 rule" from section 16.5 of *Genetic Programming III*

Residue	Increment
A, F, I, L, M, or V	0
C, D, G, H, K, N, P, Q, R, S, T, W, or Y	+2
E	

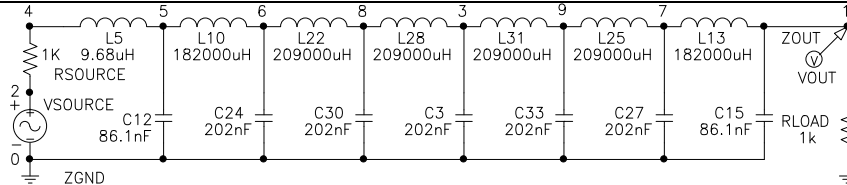
Creation of a sorting network for seven items using only 16 steps

Sections 21.4.4, 23.6, and 57.8.1 of *Genetic Programming III*



Rediscovery of the Campbell ladder topology for lowpass and highpass filters

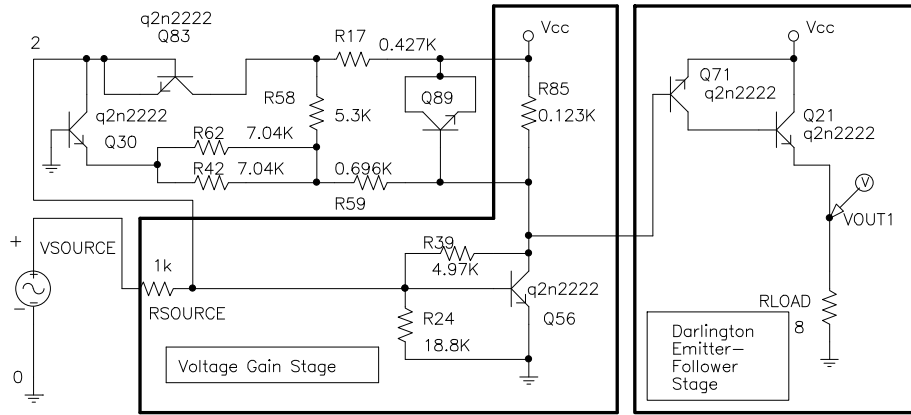
Section 25.15.1 of *Genetic Programming III*



<p>Rediscovery of the Zobel “M-derived half section” and “constant K” filter sections</p> <p>Section 25.15.2 of <i>Genetic Programming III</i></p>	
<p>Rediscovery of the Cauer (elliptic) topology for filters</p> <p>Section 27.3.7 of <i>Genetic Programming III</i></p>	
<p>Automatic decomposition of the problem of synthesizing a crossover (woofer-tweeter) filter</p> <p>Section 32.3 of <i>Genetic Programming III</i></p>	

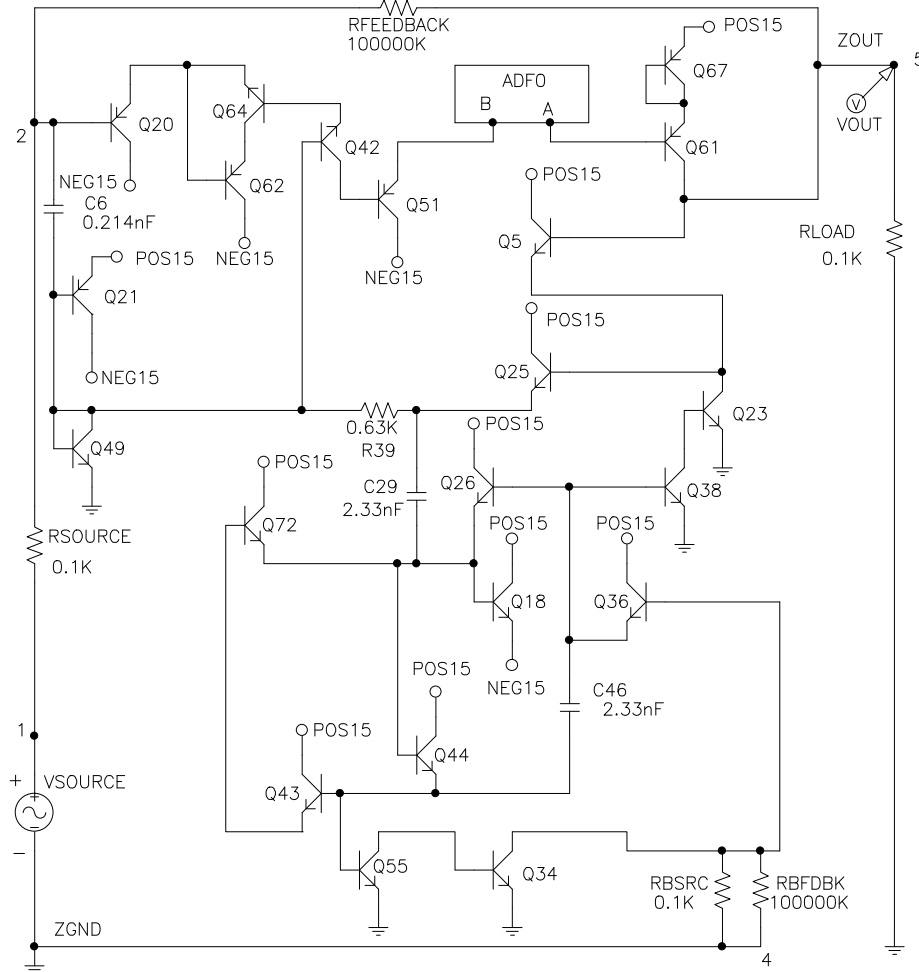
Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits

Section 42.3 of Genetic Programming III



Synthesis of 60 and 96 decibel amplifiers

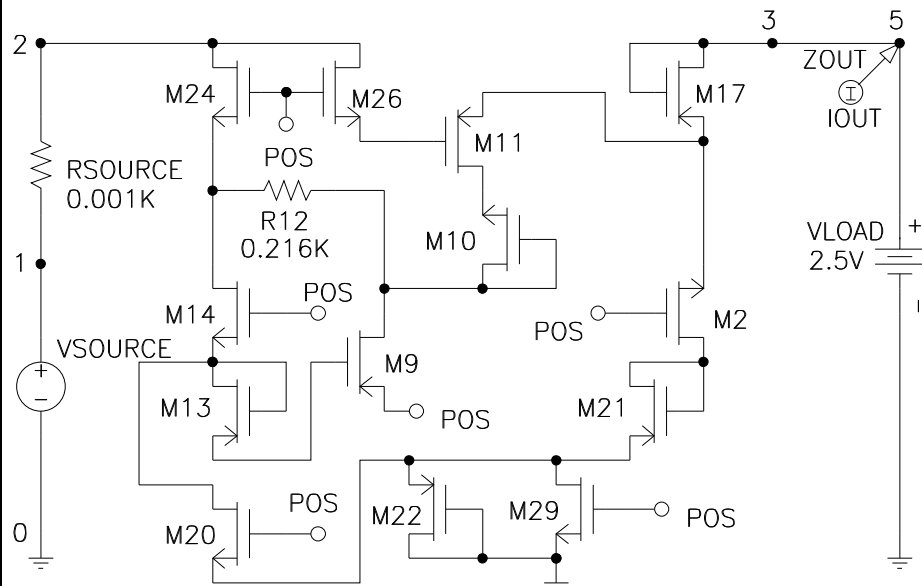
Section 45.3 of Genetic Programming III



Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions

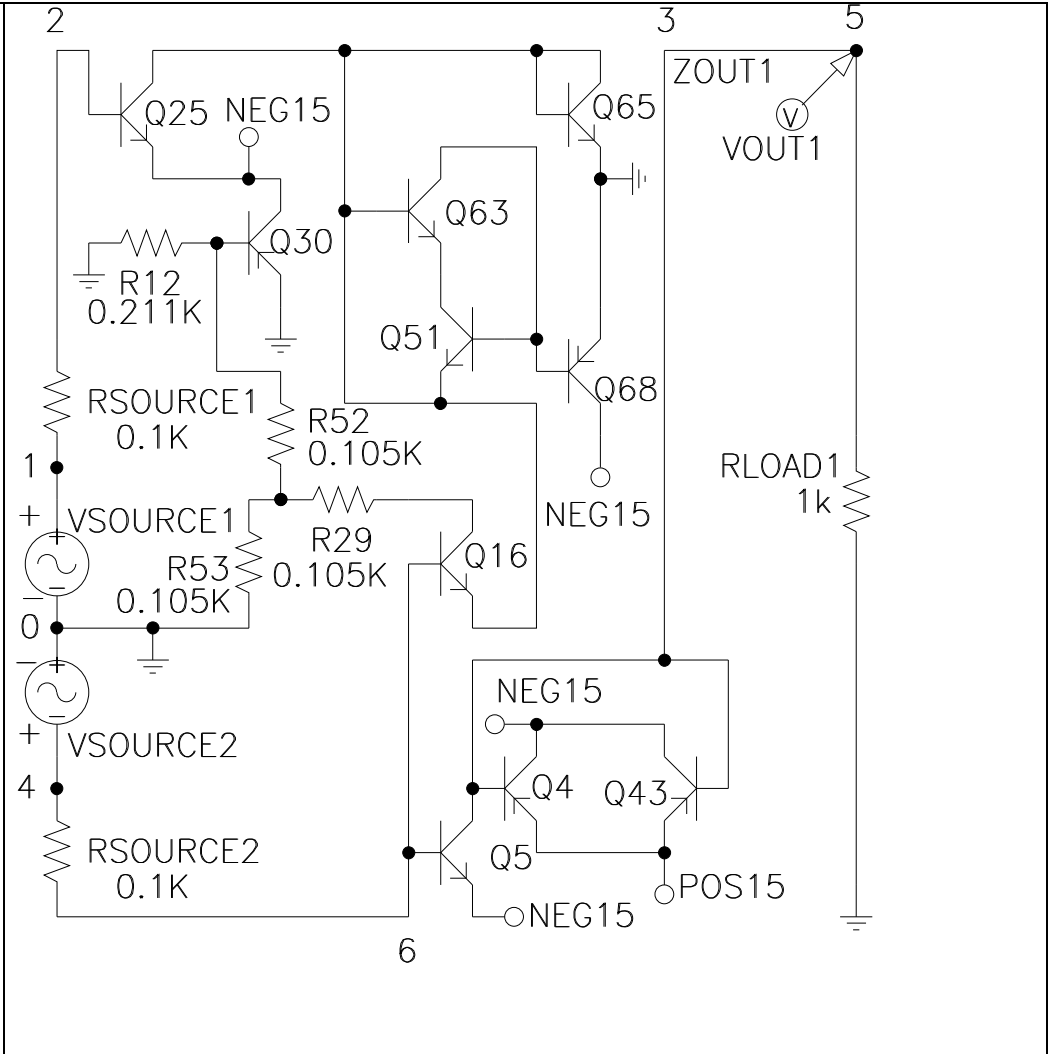
Section 47.5.3 of *Genetic Programming III*

Gaussian computational circuit using MOSFET transistors



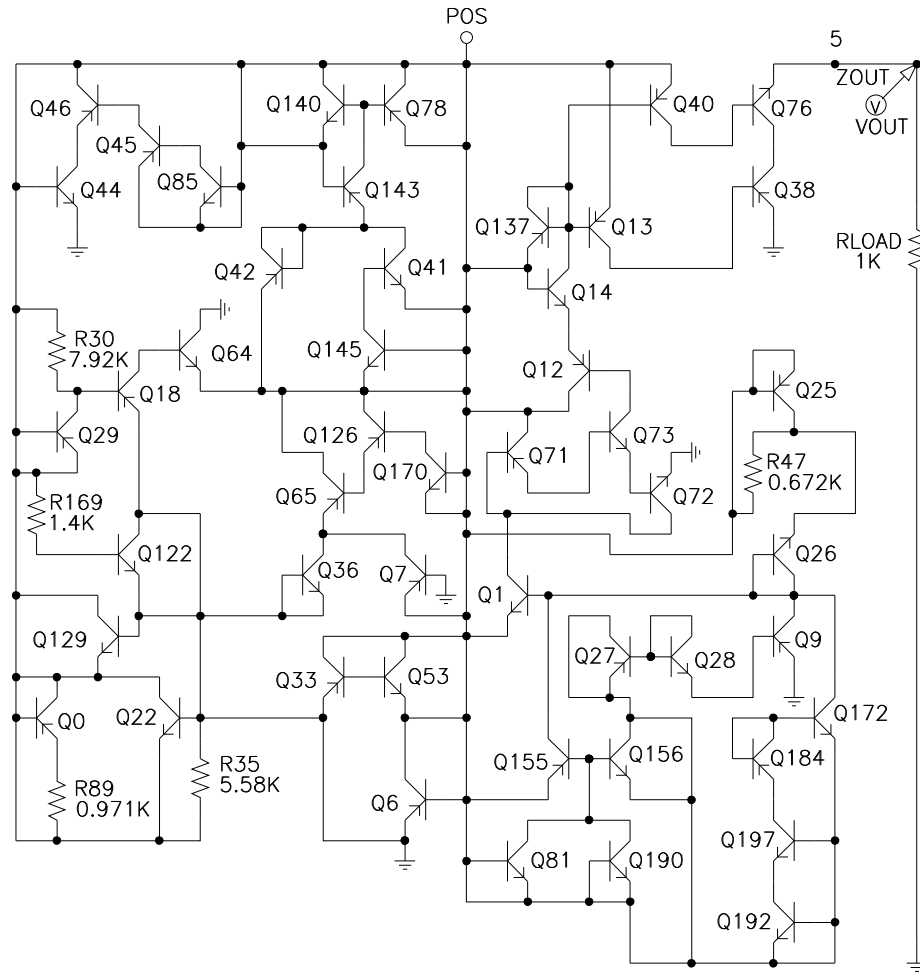
**Synthesis of a
real-time
analog circuit
for time-
optimal control
of a robot**

**Section 48.3 of
*Genetic
Programming
III***

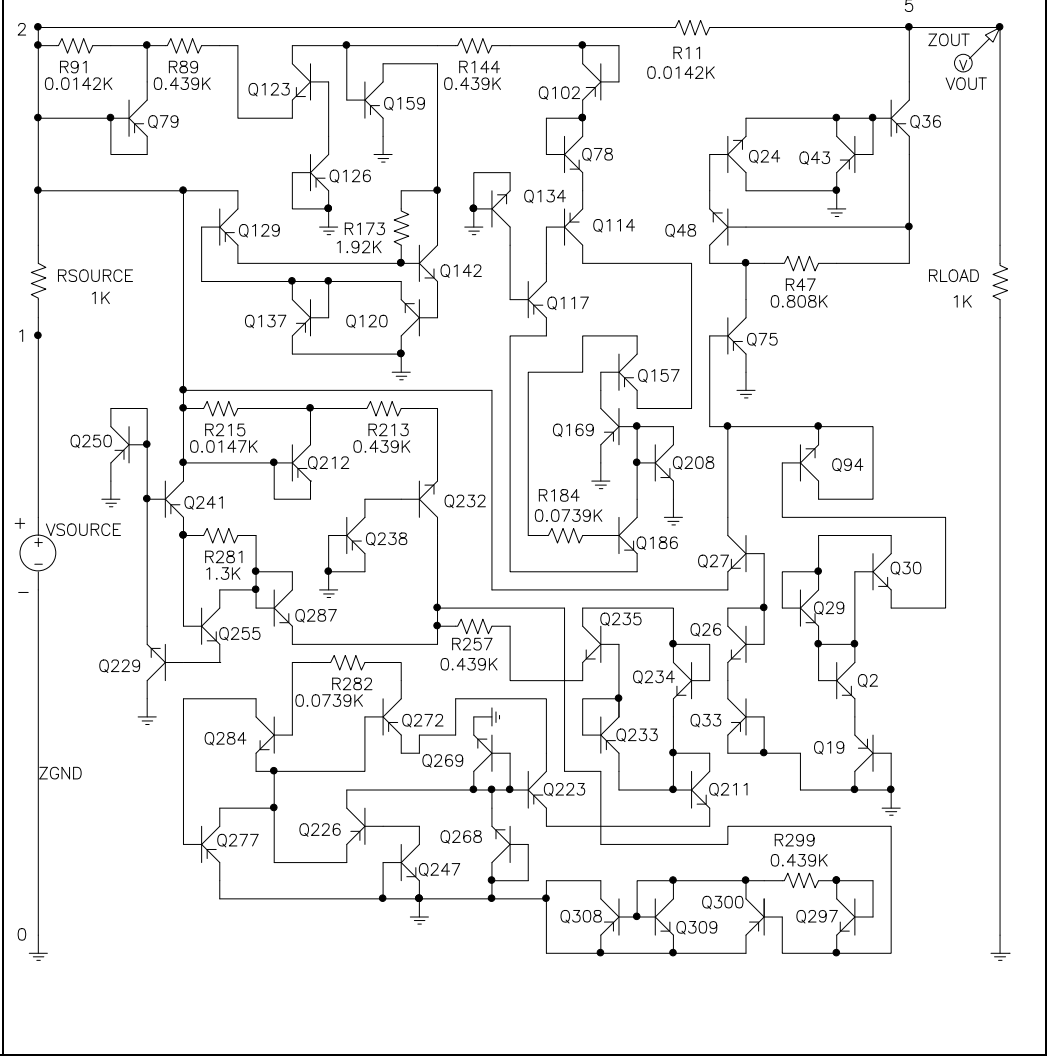


**Synthesis of an
electronic
thermometer**

**Section 49.3 of
*Genetic
Programming
III***



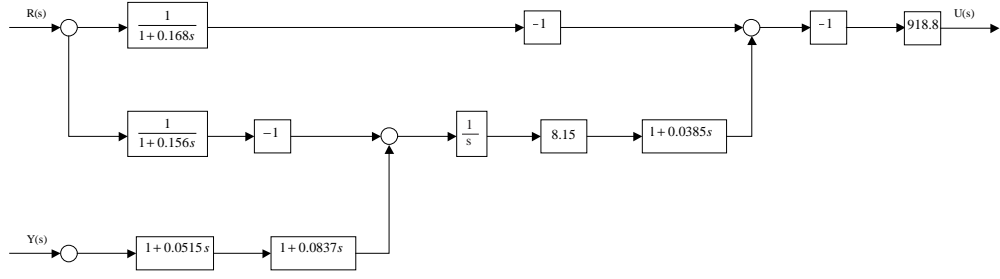
Synthesis of a voltage reference circuit
Section 50.3 of Genetic Programming III



<p>Creation of a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans</p> <p>Andre, Bennett, and Koza 1996 and section 58.4 of <i>Genetic Programming III</i></p>	<table border="1"> <thead> <tr> <th>Rule</th> <th>State Transition Rule</th> <th>Accuracy</th> </tr> </thead> <tbody> <tr> <td>Gacs-Kurdyumov-Levin (GKL) 1978 human-written</td> <td>00000000 01011111 00000000 01011111 00000000 01011111 00000000 01011111 11111111 01011111 00000000 01011111 11111111 01011111</td> <td>81.6%</td> </tr> <tr> <td>Davis 1995 human-written</td> <td>00000000 00101111 00000011 01011111 00000000 00011111 11001111 00011111 00000000 00101111 11111100 01011111 00000000 00011111 11111111 00011111</td> <td>81.800%.</td> </tr> <tr> <td>Das (1995) human-written</td> <td>00000111 00000000 00000111 11111111 00001111 00000000 00001111 11111111 00001111 00000000 00000111 11111111 00001111 00110001 00001111 11111111</td> <td>82.178%</td> </tr> <tr> <td>Best rule evolved by genetic programming (1999)</td> <td>00000101 00000000 01010101 00000101 00000101 00000000 01010101 00000101 01010101 11111111 01010101 11111111 01010101 11111111 01010101 11111111</td> <td>82.326%</td> </tr> </tbody> </table>	Rule	State Transition Rule	Accuracy	Gacs-Kurdyumov-Levin (GKL) 1978 human-written	00000000 01011111 00000000 01011111 00000000 01011111 00000000 01011111 11111111 01011111 00000000 01011111 11111111 01011111	81.6%	Davis 1995 human-written	00000000 00101111 00000011 01011111 00000000 00011111 11001111 00011111 00000000 00101111 11111100 01011111 00000000 00011111 11111111 00011111	81.800%.	Das (1995) human-written	00000111 00000000 00000111 11111111 00001111 00000000 00001111 11111111 00001111 00000000 00000111 11111111 00001111 00110001 00001111 11111111	82.178%	Best rule evolved by genetic programming (1999)	00000101 00000000 01010101 00000101 00000101 00000000 01010101 00000101 01010101 11111111 01010101 11111111 01010101 11111111 01010101 11111111	82.326%
	Rule	State Transition Rule	Accuracy													
	Gacs-Kurdyumov-Levin (GKL) 1978 human-written	00000000 01011111 00000000 01011111 00000000 01011111 00000000 01011111 11111111 01011111 00000000 01011111 11111111 01011111	81.6%													
	Davis 1995 human-written	00000000 00101111 00000011 01011111 00000000 00011111 11001111 00011111 00000000 00101111 11111100 01011111 00000000 00011111 11111111 00011111	81.800%.													
	Das (1995) human-written	00000111 00000000 00000111 11111111 00001111 00000000 00001111 11111111 00001111 00000000 00000111 11111111 00001111 00110001 00001111 11111111	82.178%													
Best rule evolved by genetic programming (1999)	00000101 00000000 01010101 00000101 00000101 00000000 01010101 00000101 01010101 11111111 01010101 11111111 01010101 11111111 01010101 11111111	82.326%														
<p>Creation of motifs that detect the D-E-A-D box family of proteins and the manganese superoxide dismutase family</p> <p>Section 59.8 of <i>Genetic Programming III</i></p>	<p>[IV] - [lim] - D - E - [AI] - D - [rnek] - [lim] - [lim] - [limeqdnrsk]</p>															

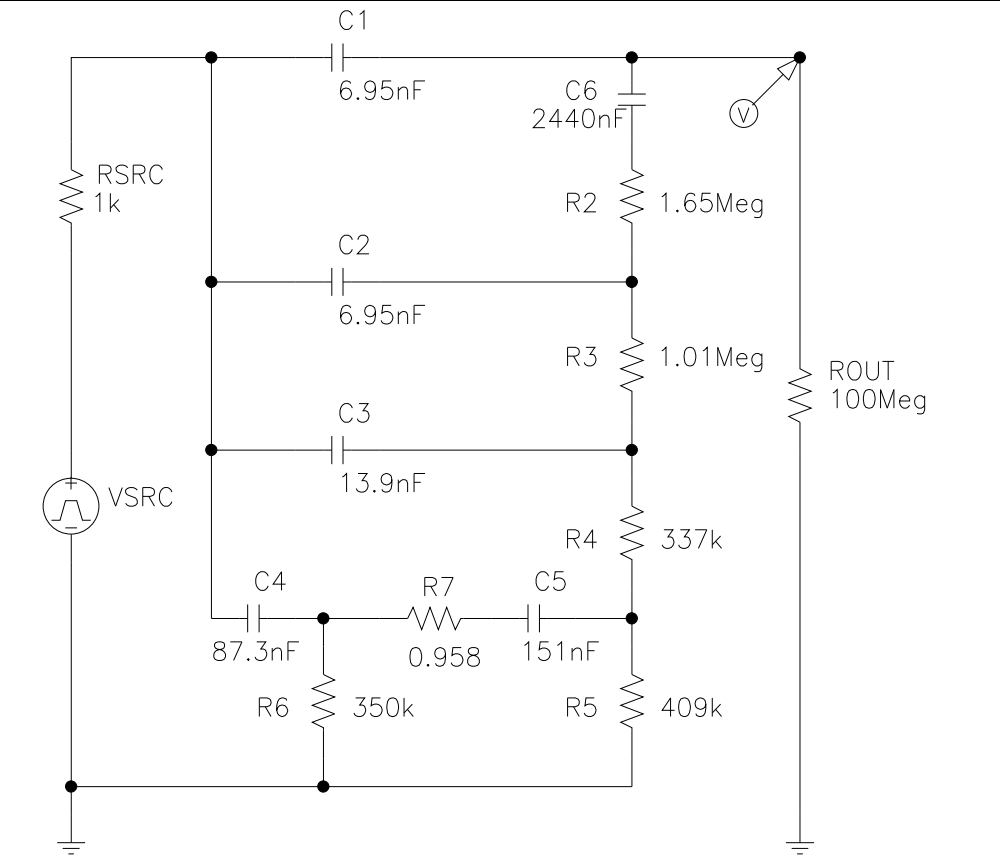
Synthesis of topology for a PID-D2 (proportional, integrative, derivative, and second derivative) controller

Section 3.7 of Genetic Programming IV



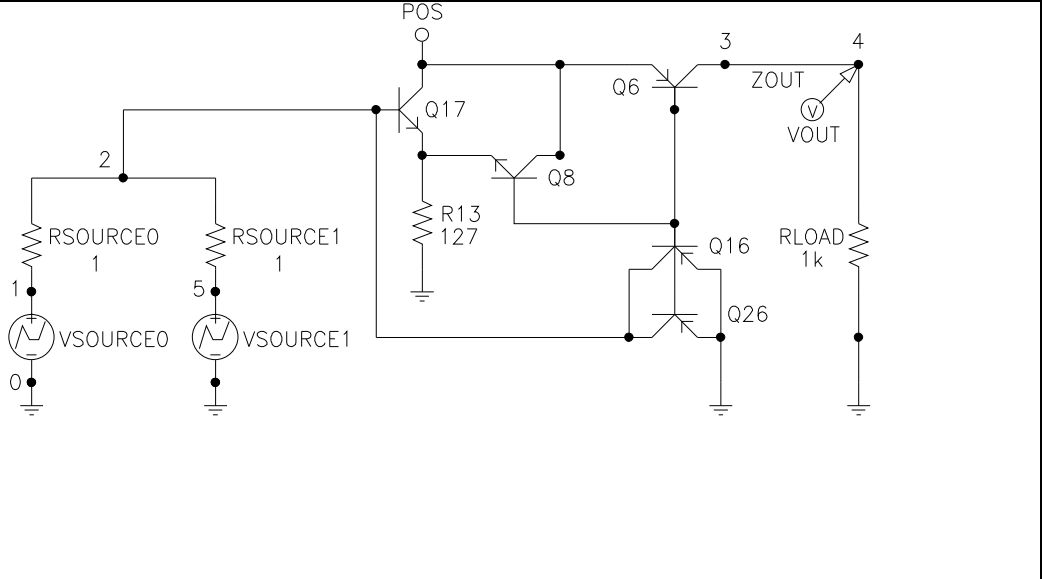
Synthesis of an analog circuit equivalent to Philbrick circuit

Section 4.3 of Genetic Programming IV

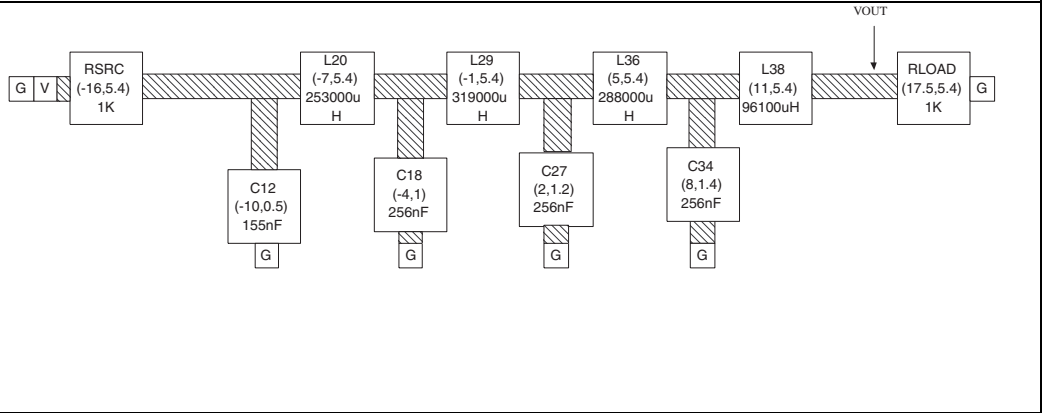


Synthesis of a NAND circuit

Section 4.4 of *Genetic Programming IV*

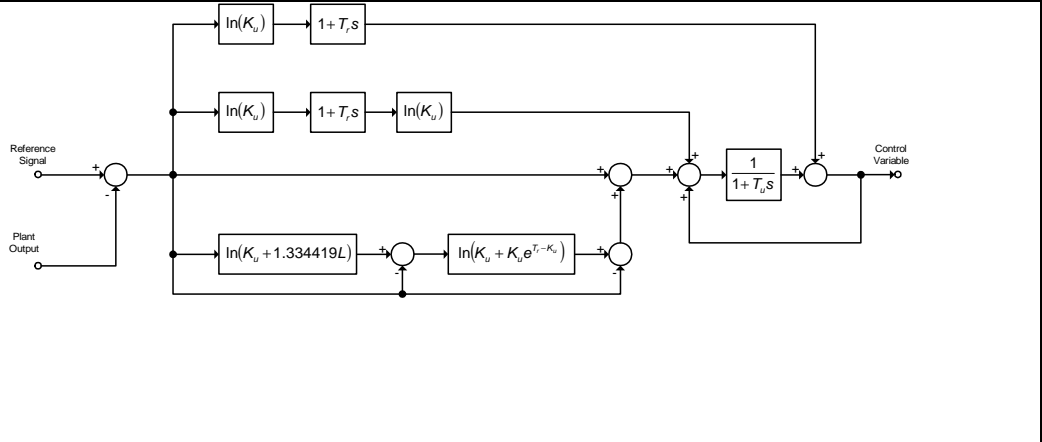


Simultaneous synthesis of topology, sizing, placement, and routing of analog electrical circuits



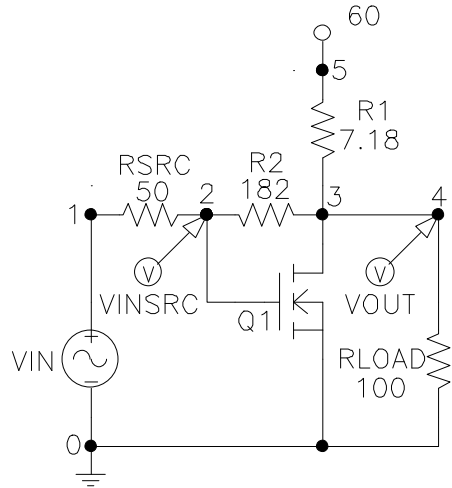
Synthesis of topology for a PID (proportional, integrative, and derivative) controller

Chapter 5 of *Genetic Programming IV*



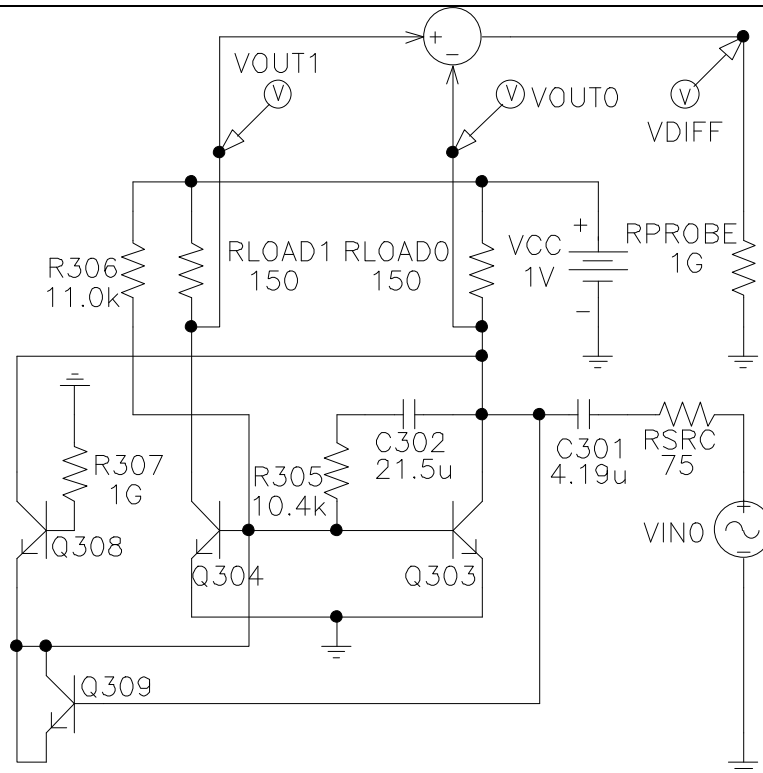
**Rediscovery of
negative
feedback**

**Chapter 14 of
*Genetic
Programming
IV***



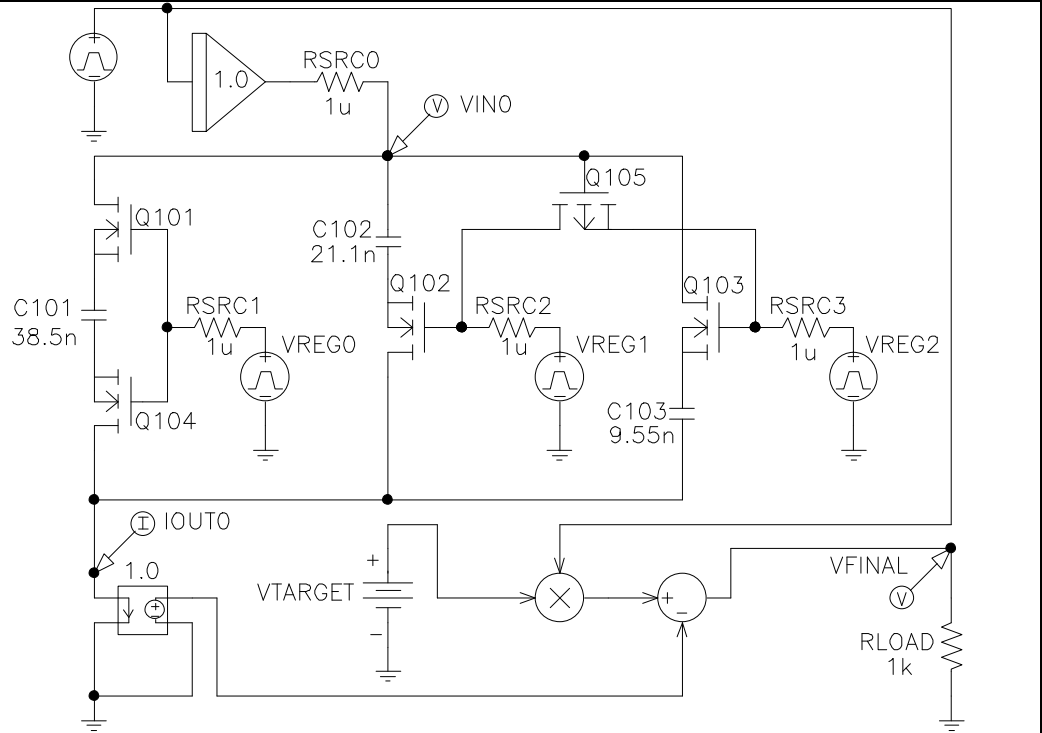
**Synthesis of a
low-voltage
balun circuit**

**Section 15.4.1
of *Genetic
Programming
IV***



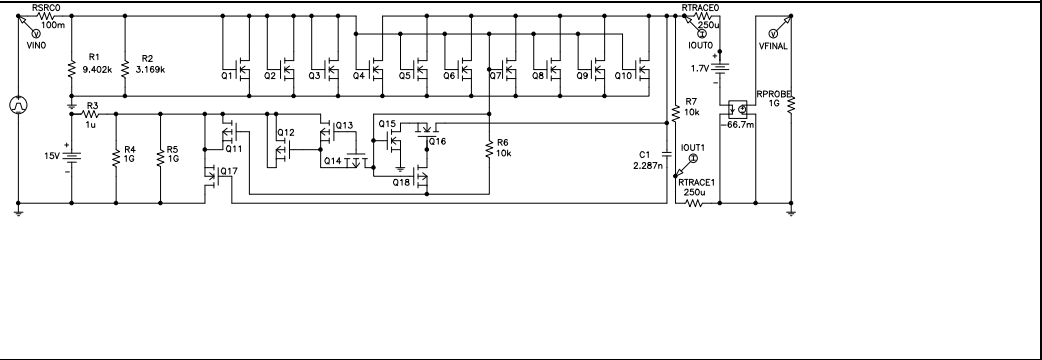
Synthesis of a mixed analog-digital variable capacitor circuit

Section 15.4.2 of Genetic Programming IV



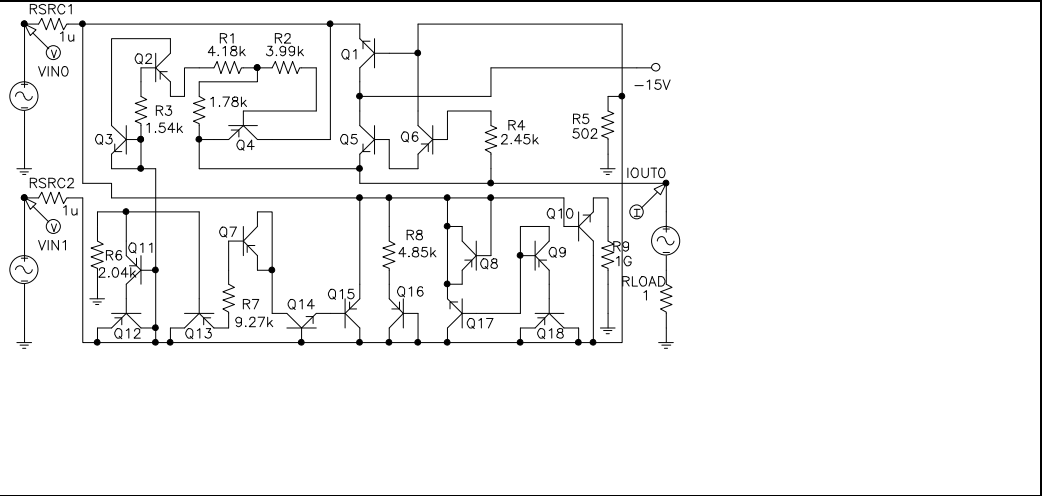
Synthesis of a high-current load circuit

Section 15.4.3 of Genetic Programming IV



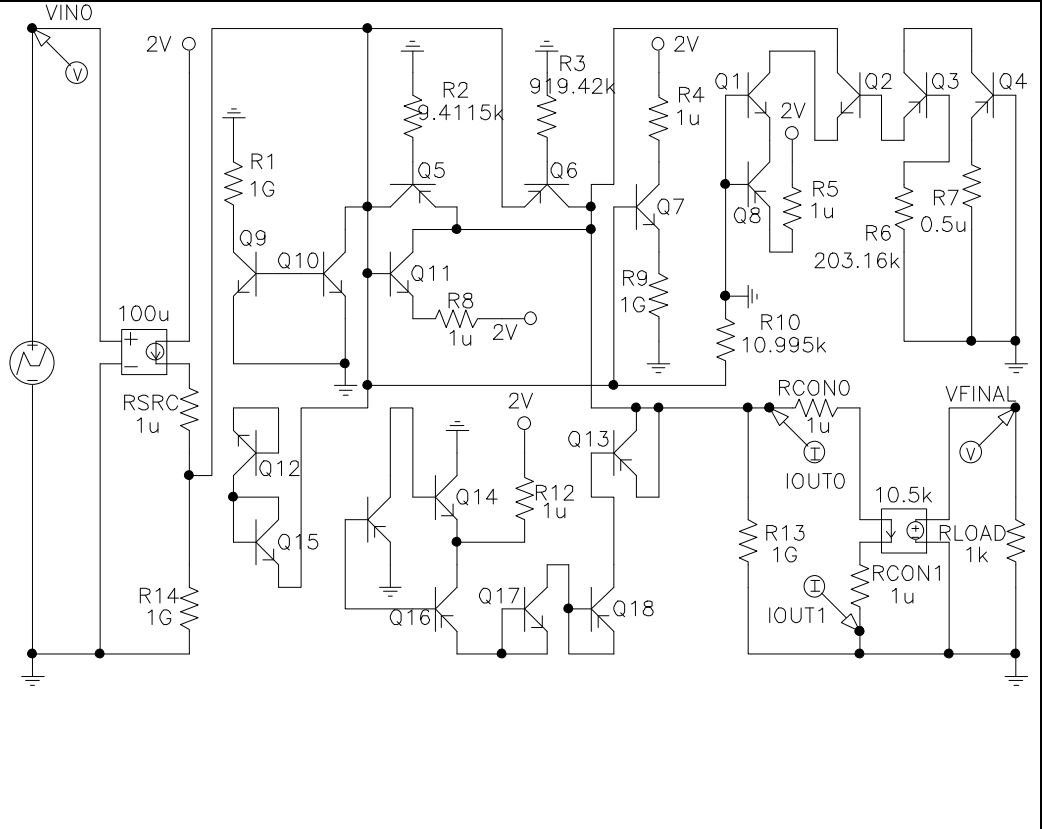
Synthesis of a voltage-current conversion circuit

Section 15.4.4 of Genetic Programming IV



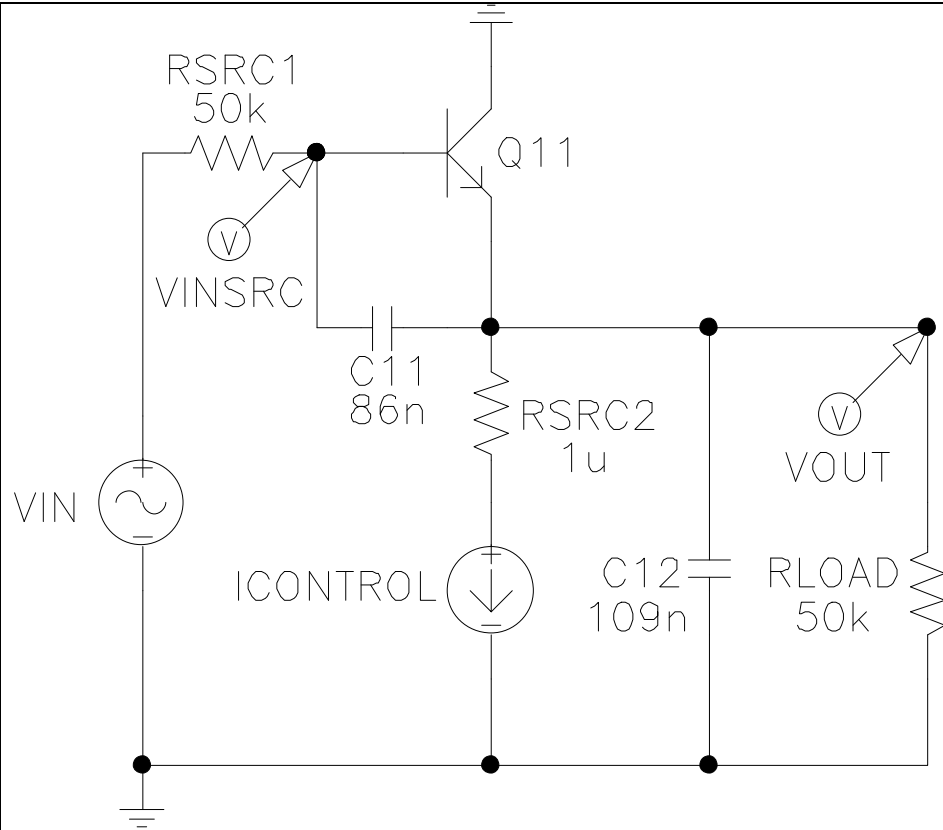
Synthesis of a cubic function generator

Section 15.4.5 of Genetic Programming IV



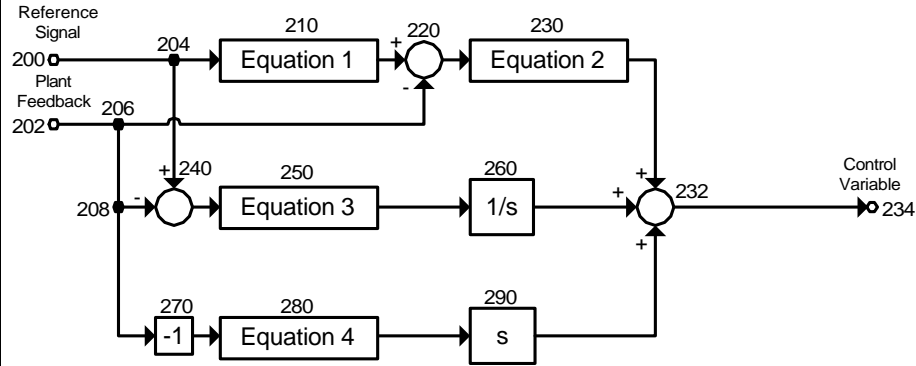
Synthesis of a
tunable
integrated
active filter

Section 15.4.6
of *Genetic
Programming
IV*



Creation of PID tuning rules that outperform the Ziegler-Nichols and Åström-Hägglund tuning rules

Chapter 12 of Genetic Programming IV



The topology (above) was not evolved, but was the standard PID topology. Evolved equations for $K_{p-final}$, $K_{i-final}$, $K_{d-final}$, and b_{final} :

$$K_{p-final} = 0.72 * K_u * e^{\frac{-1.6}{K_u} + \frac{1.2}{K_u^2}} - .0012340 * T_u - 6.1173 * 10^{-6}$$

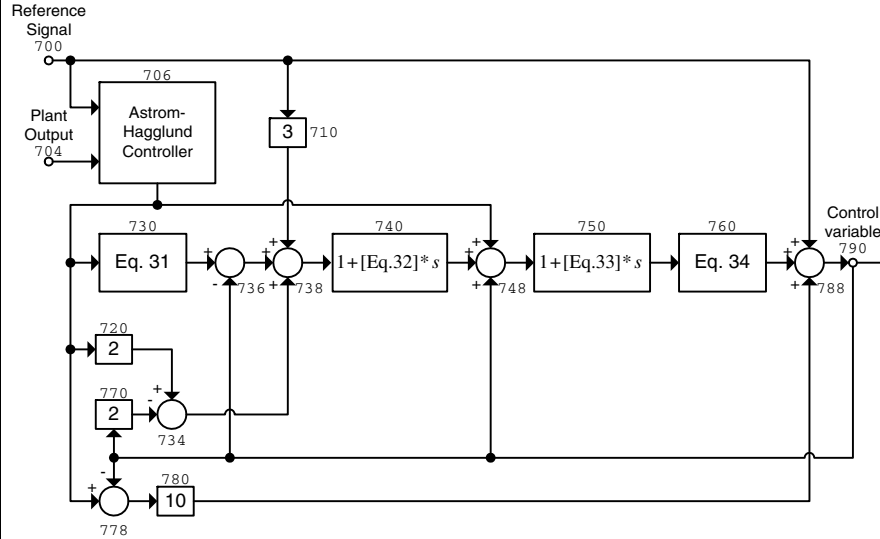
$$K_{i-final} = \frac{0.72 * K_u * e^{\frac{-1.6}{K_u} + \frac{1.2}{K_u^2}} - .068525 * \frac{K_u}{T_u}}{0.59 * T_u * e^{\frac{-1.3}{K_u} + \frac{0.38}{K_u^2}}}$$

$$K_{d-final} = 0.108 * K_u * T_u * e^{\frac{-1.6}{K_u} + \frac{1.2}{K_u^2}} * e^{\frac{-1.4}{K_u} + \frac{0.56}{K_u^2}} - 0.0026640 (e^{T_u})^{\log(1.6342 \log K_u)}$$

$$b_{final} = 0.25 * e^{\frac{0.56}{K_u} + \frac{-0.12}{K_u^2}} + \frac{K_u}{e^{K_u}}$$

Creation of three non-PID controllers that outperform a PID controller that uses the Ziegler-Nichols or Åström-Hägglund tuning rules

Chapter 13 of Genetic Programming IV



The above topology and equations 31, 32, 33, and 34 were evolved:

$$\left\| \log \left| T_r - T_u + \log \frac{\log(|L|^L)}{T_u + 1} \right| \right\| \quad [31]$$

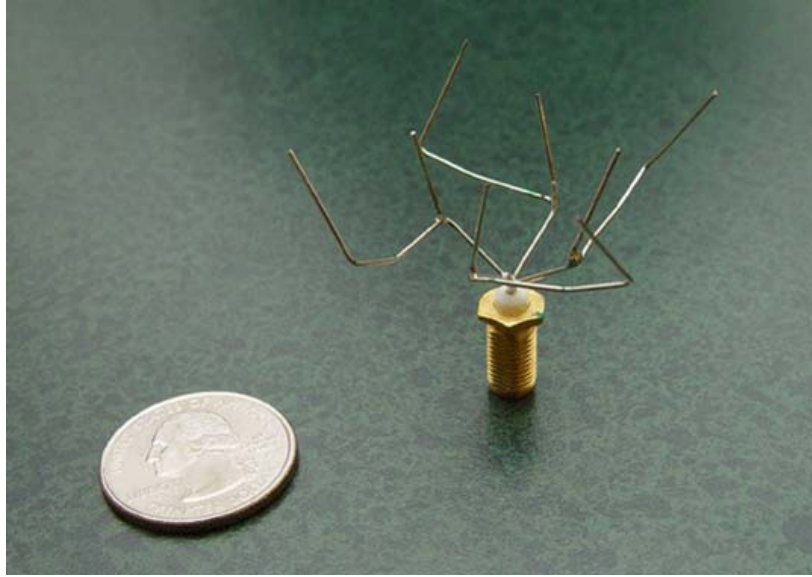
$$\left\| \log |T_r + 1| \right\| \quad [34]$$

$$NLM \left(\log |L| - (\text{abs}(L)^L)^2 T_u^3 (T_u + 1) T_r e^L - 2T_u e^L \right) \quad [32]$$

$$NLM \left(\log |L| - 2T_u e^L \left(2K_u \left(\log |K_u e^L| - \log |L| \right) T_u + K_u e^L \right) \right) \quad [33]$$

**Antenna that
satisfied NASA
specs and that
will be
launched into
space in 2004**

Lohn et al. 2003

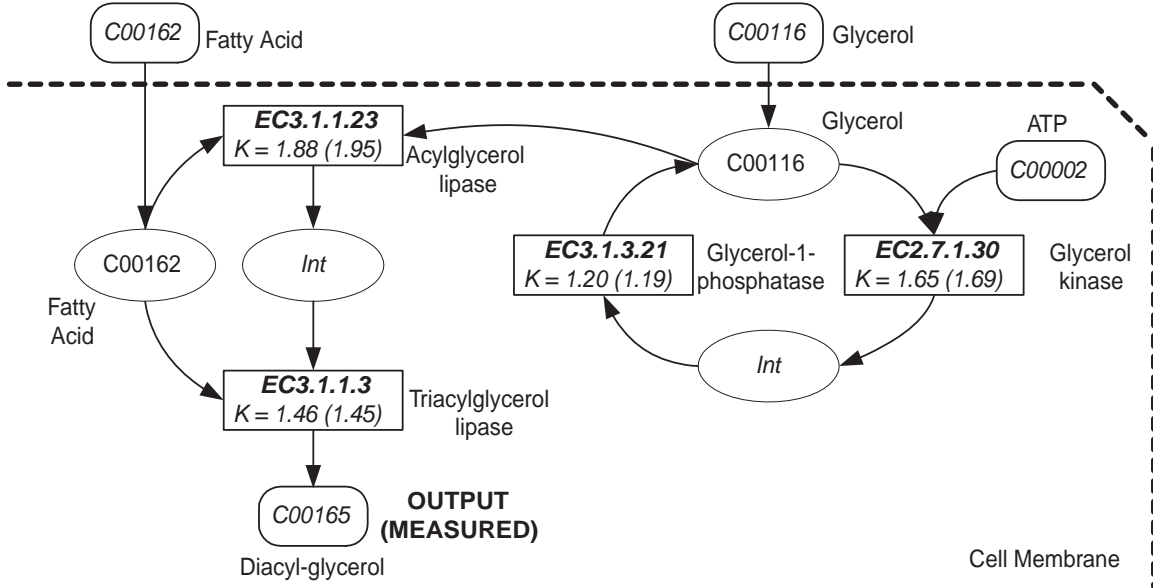


**ANNUAL HUMAN-COMPETITIVE
AWARDS (“HUMIES”)**

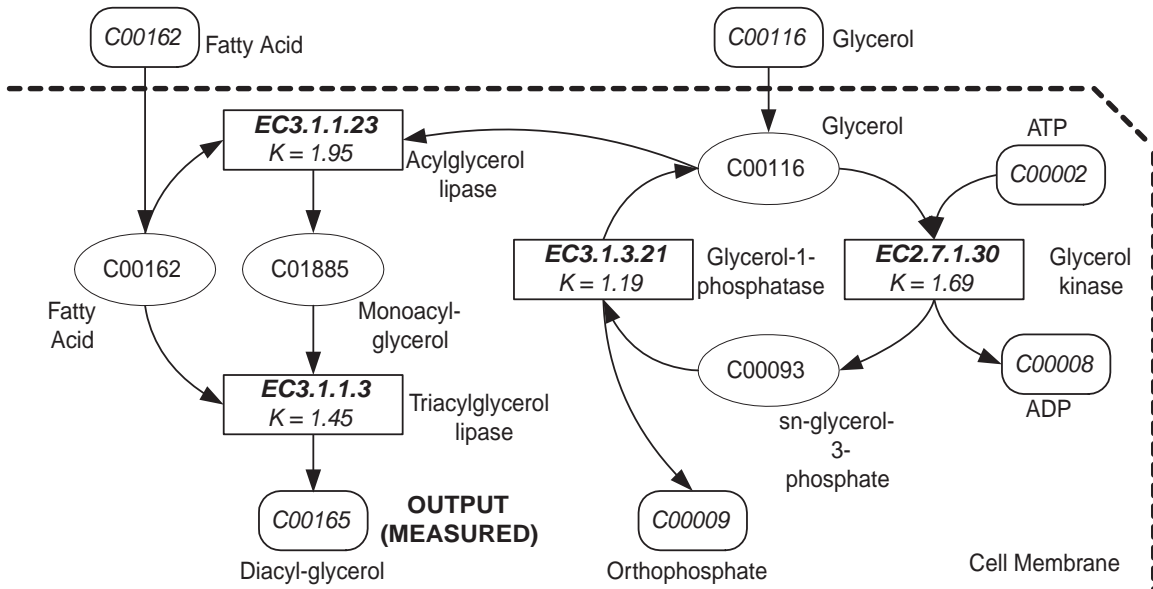
www.Human-Competitive.org

REVERSE ENGINEERING OF METABOLIC PATHWAYS (4-REACTION NETWORK IN PHOSPHOLIPID CYCLE)

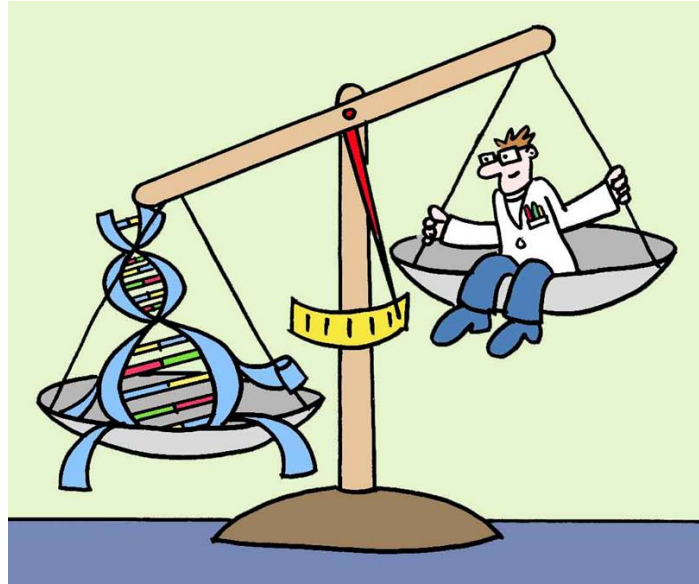
BEST-OF-GENERATION 66



DESIRED

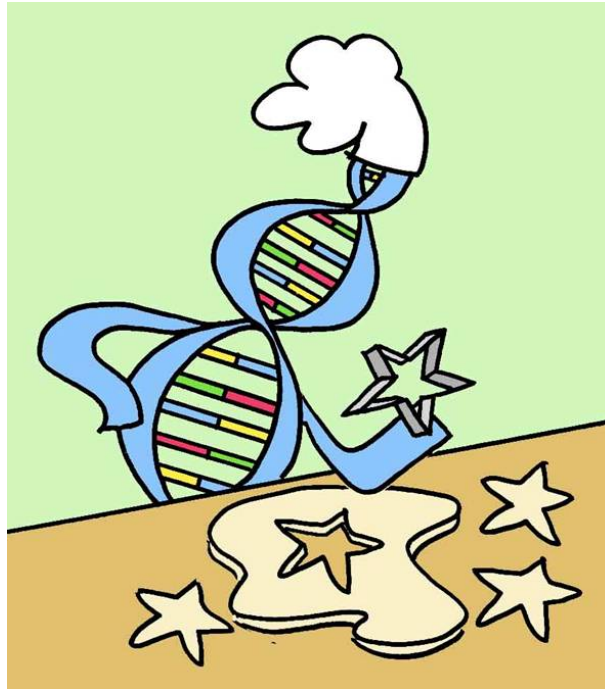


DEFINITION OF “HIGH-RETURN” BASED ON THE “AI RATIO”



The *AI ratio* (the “artificial-to-intelligence” ratio) of a problem-solving method as the ratio of that which is delivered by the automated operation of the *artificial* method to the amount of *intelligence* that is supplied by the human applying the method to a particular problem.

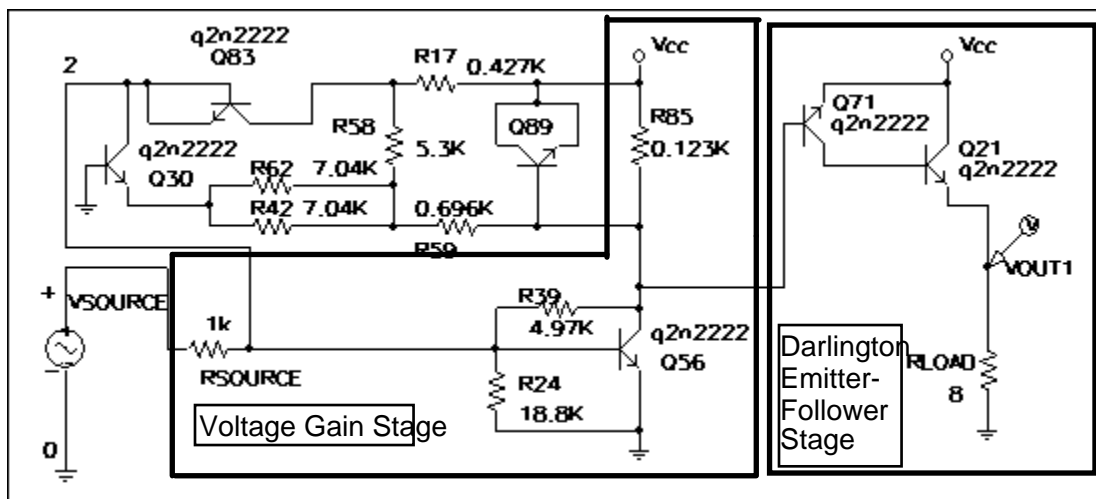
DEFINITION OF “ROUTINE”



A problem-solving method is *routine* if it is general and relatively little human effort is required to get the method to successfully handle new problems within a particular domain and to successfully handle new problems from a different domain.

GENETICALLY EVOLVED 10 DB AMPLIFIER FROM GENERATION 45

SHOWING THE VOLTAGE GAIN STAGE AND DARLINGTON EMITTER FOLLOWER SECTION



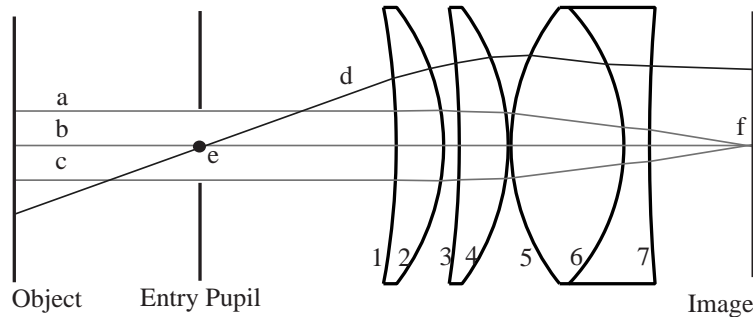
**CROSS-DOMAIN OBSERVATIONS
ABOUT RUNS OF GENETIC
PROGRAMMING USED TO
AUTOMATICALLY CREATE DESIGNS
FOR ANALOG CIRCUITS, OPTICAL
LENS SYSTEMS, CONTROLLERS,
ANTENNAS, MECHANICAL SYSTEMS,
AND QUANTUM COMPUTING CIRCUITS**

- **optical lens systems (Al-Sakran, Koza, and Jones, 2005; Koza, Al-Sakran, and Jones, 2005),**
- **antennas (Lohn, Hornby, and Linden 2004; Comisky, Yu, and Koza 2000),**
- **analog electrical circuits (Koza, Bennett, Andre, and Keane 1996; Koza, Bennett, Andre, and Keane 1999),**
- **controllers (Koza, Keane, Streeter, Mydlowec, Yu, and Lanza 2003; Keane, Koza, Streeter 2005),**
- **mechanical systems (Lipson 2004), and**
- **quantum computing circuits (Spector 2004)**

CROSS-DOMAIN FEATURES

- **Native representations are sufficient when working with genetic programming**
- **Genetic programming breeds simulatability**
- **Genetic programming starts small**
- **Genetic programming frequently exploits a simulator's built-in assumption of reasonableness**
- **Genetic programming engineers around existing patents and creates novel designs more frequently than it creates infringing solutions**

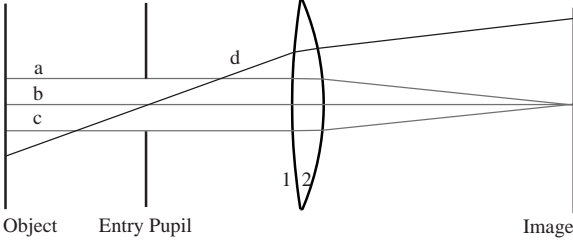
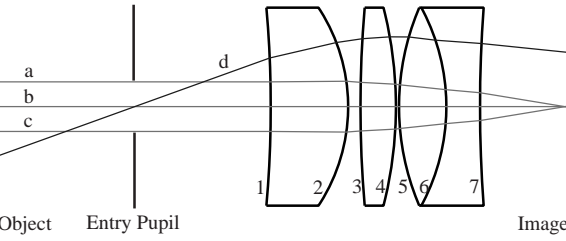
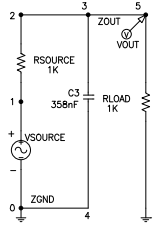
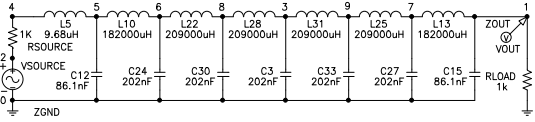
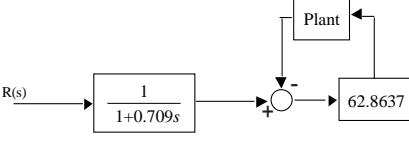
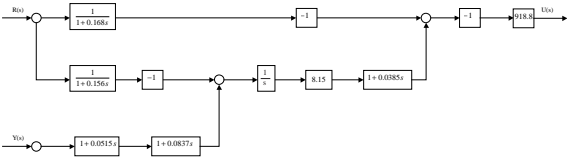
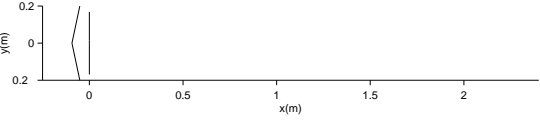
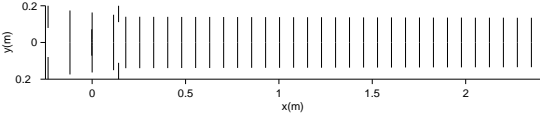
NATIVE REPRESENTATIONS ARE USUALLY SUFFICIENT WHEN WORKING WITH GENETIC PROGRAMMING

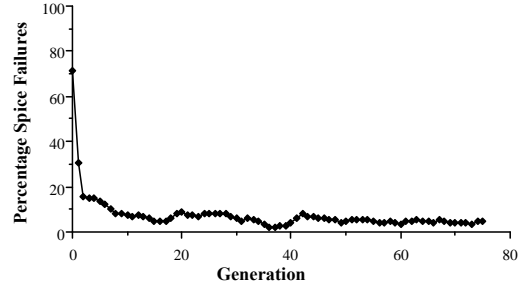


“PRESCRIPTION” (“LENS FILE”)

Surface	Distance	Radius	Material	Aperture
Object	10^{10}	flat	air	
Entry pupil	0.88	flat	air	0.18
1	0.21900	-3.5236	BK7	0.62
2	0.07280	-1.0527	air	0.62
3	0.22500	-4.4072	BK7	0.62
4	0.01360	-1.0704	air	0.62
5	0.52100	1.02491	BK7	0.62
6	0.11800	-0.9349	SF61	0.62
7	0.47485	7.94281	air	0.62
Image		flat		

GP STARTS SMALL

Best-of-generation 0	Best-of-run
 <p>Object Entry Pupil Image</p> <p>Optical lens system</p>	 <p>Object Entry Pupil Image</p> <p>Optical lens system</p>
 <p>Lowpass filter</p>	 <p>Lowpass filter</p>
 <p>Controller</p>	 <p>Controller</p>
 <p>Antenna</p>	 <p>Antenna</p>



GENETIC PROGRAMMING BREEDS SIMULATABILITY

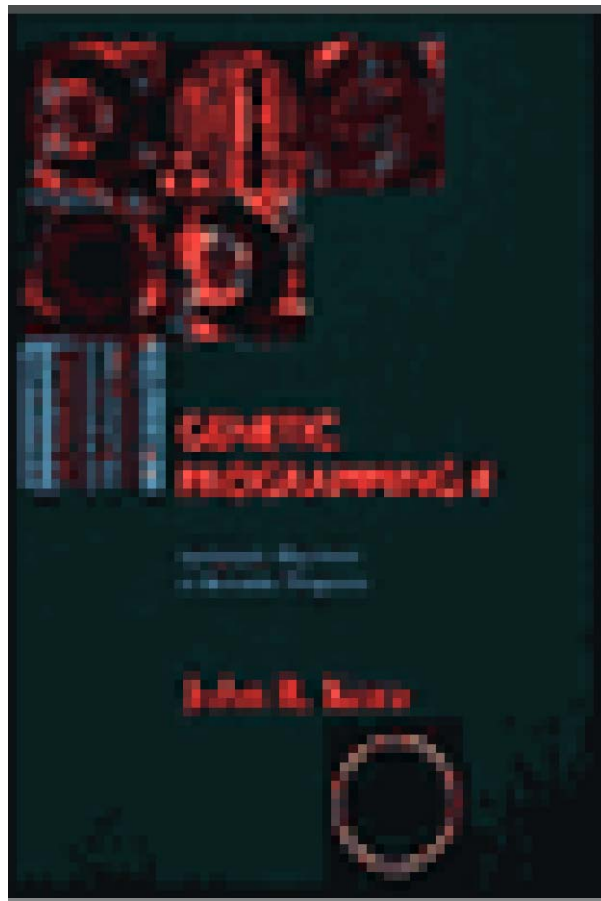
Unsimulatable individuals

**GENETIC PROGRAMMING ENGINEERS
AROUND EXISTING PATENTS AND
CREATES NOVEL DESIGNS MORE
FREQUENTLY THAN IT CREATES
INFRINGEMENT SOLUTIONS**

**GENETIC PROGRAMMING
FREQUENTLY EXPLOITS A
SIMULATOR'S BUILT-IN ASSUMPTION
OF REASONABLENESS**

AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

**8 MAIN POINTS FROM BOOK
*GENETIC PROGRAMMING II:
AUTOMATIC DISCOVERY OF REUSABLE
PROGRAMS* (KOZA 1994)**



AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

- **ADFs work.**
- **ADFs do not solve problems in the style of human programmers.**
- **ADFs reduce the computational effort required to solve a problem.**
- **ADFs usually improve the parsimony of the solutions to a problem.**
- **As the size of a problem is scaled up, the size of solutions increases more slowly with ADFs than without them.**
- **As the size of a problem is scaled up, the computational effort required to solve a problem increases more slowly with ADFs than without them.**
- **The advantages in terms of computational effort and parsimony conferred by ADFs increase as the size of the problem is scaled up.**

REUSE

AUTOMATICALLY DEFINED ITERATIONS (ADIs)

- Overall program consisting of an automatically defined function **ADF0**, an iteration-performing branch **IPB0**, and a result-producing branch **RPB0**.
- Iteration is over a known, fixed set
 - protein or DNA sequence (of varying length)
 - time-series data
 - two-dimensional array of pixels

REUSE—TRANSMEMBRANE SEGMENT IDENTIFICATION PROBLEM

- **Goal is to classify a given protein segment as being a transmembrane domain or non-transmembrane area of the protein**
- **Generation 20 — Run 3 — Subset-creating version**
 - **in-sample correlation of 0.976**
- **After cross-validation**
 - **out-of-sample correlation of 0.968**
 - **out-of-sample error rate 1.6%**

REUSE—TRANSMEMBRANE SEGMENT IDENTIFICATION PROBLEM

```
(progn
```

```
  (defun ADF0 ()
    (ORN (ORN (ORN (I?) (H?)) (ORN (P?) (G?))) (ORN (ORN
      (ORN (Y?) (N?)) (ORN (T?) (Q?))) (ORN (A?) (H?))))))
```

```
  (defun ADF1 ()
    (values (ORN (ORN (ORN (A?) (I?)) (ORN (L?) (W?)))
      (ORN (ORN (T?) (L?)) (ORN (T?) (W?))))))
```

```
  (defun ADF2 ()
    (values (ORN (ORN (ORN (ORN (ORN (D?) (E?)) (ORN (ORN
      (ORN (D?) (E?)) (ORN (ORN (T?) (W?)) (ORN (Q?)
      (D?)))) (ORN (K?) (P?)))) (ORN (K?) (P?)) (ORN (T?)
      (W?))) (ORN (ORN (E?) (A?)) (ORN (N?) (R?))))))
```

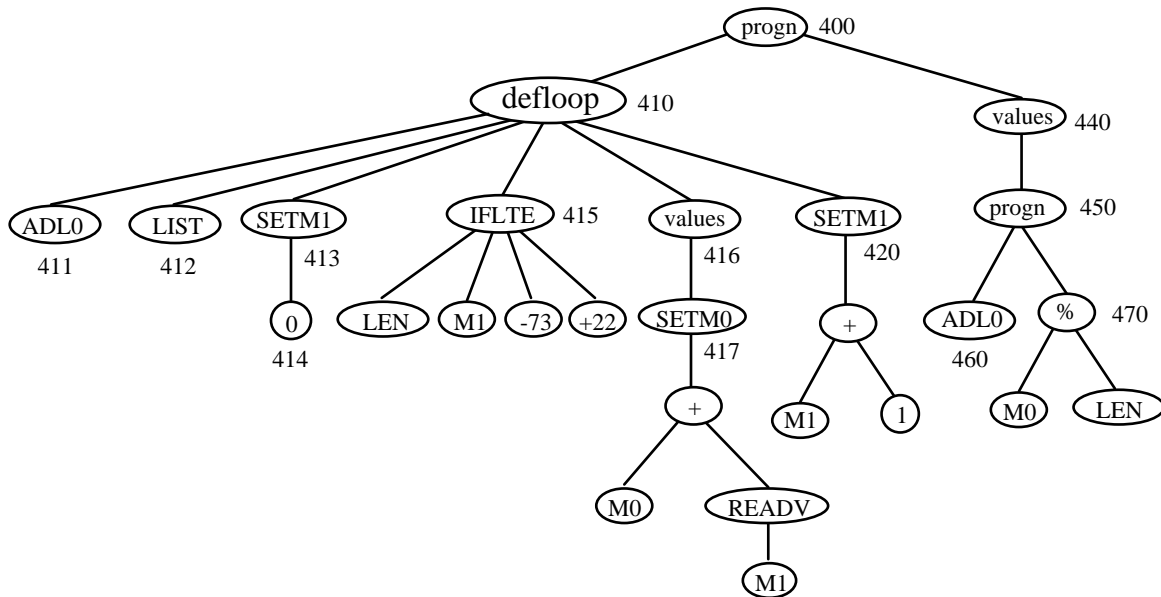
```
  (progn (loop-over-residues
    (SETM0 (+ (- (ADF1) (ADF2)) (SETM3 M0))))
```

```
  (values (% (% M3 M0) (% (% (% (- L -0.53) (* M0
    M0)) (+ (% (% M3 M0) (% (+ M0 M3) (% M1 M2))) M2)) (%
    M3 M0))))))
```

- GP created the body of 3 subroutines (ADFs), 1 iteration-performing branch, and 1 result-producing branch (RPB) were created by genetic programming

REUSE

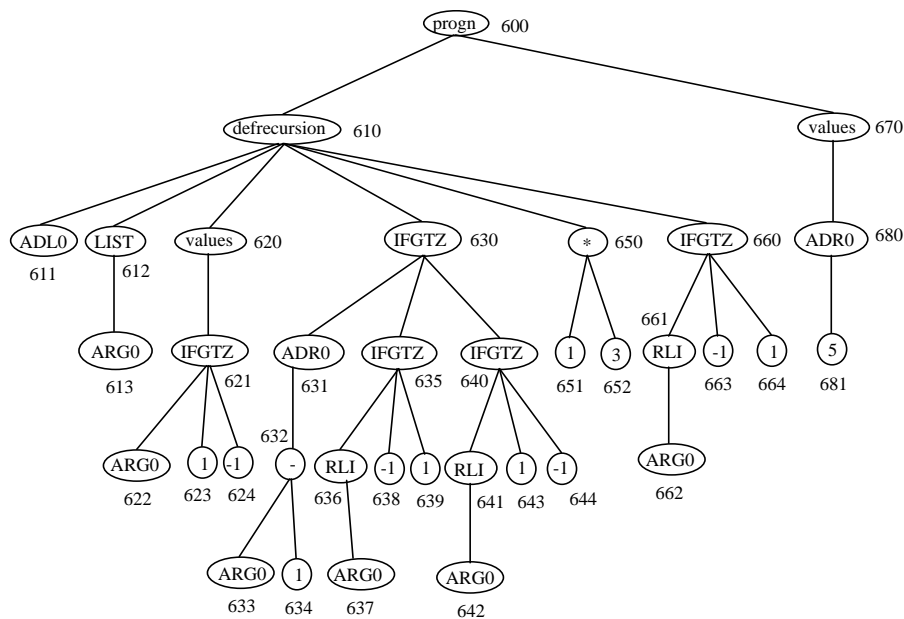
EXAMPLE OF A PROGRAM WITH A FOUR-BRANCH AUTOMATICALLY DEFINED LOOP (ADL0) AND A RESULT- PRODUCING BRANCH



REUSE

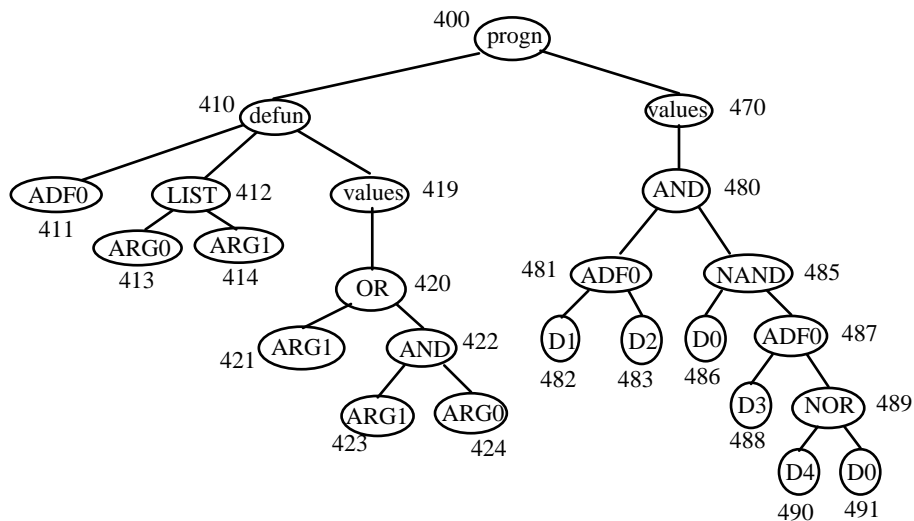
AUTOMATICALLY DEFINED RECURSION (ADRO) AND A RESULT- PRODUCING BRANCH

- a recursion condition branch, RCB
- a recursion body branch, RBB
- a recursion update branch, RUB
- a recursion ground branch, RGB



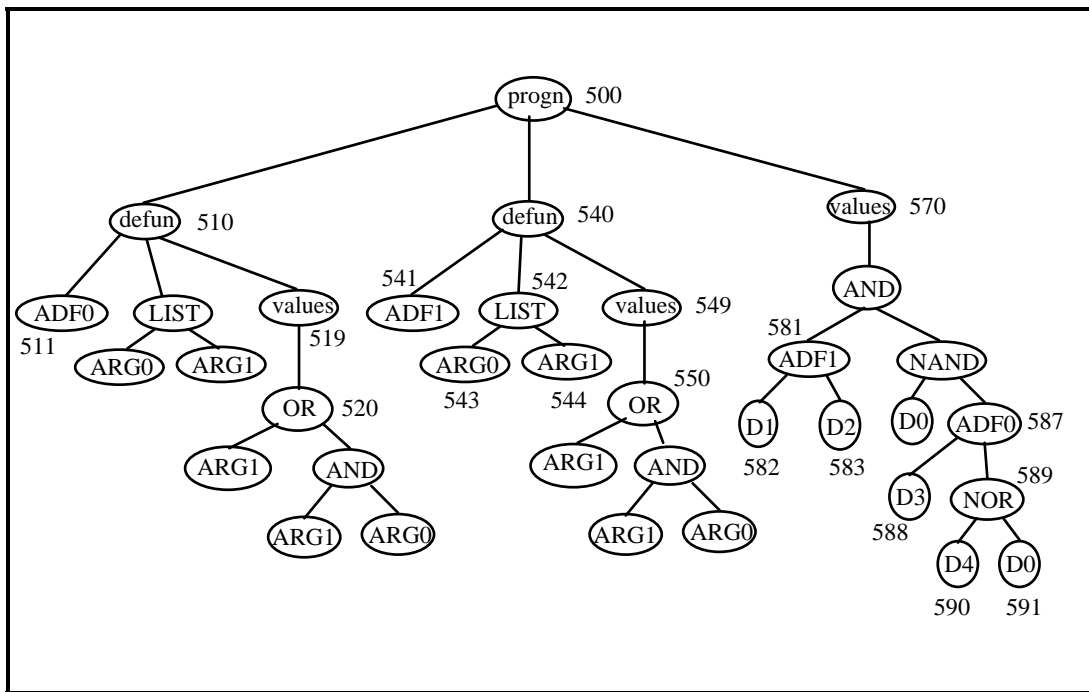
ARCHITECTURE-ALTERING OPERATIONS

PROGRAM WITH 1 TWO-ARGUMENT AUTOMATICALLY DEFINED FUNCTION (ADF0) AND 1 RESULT-PRODUCING BRANCH – ARGUMENT MAP OF {2}



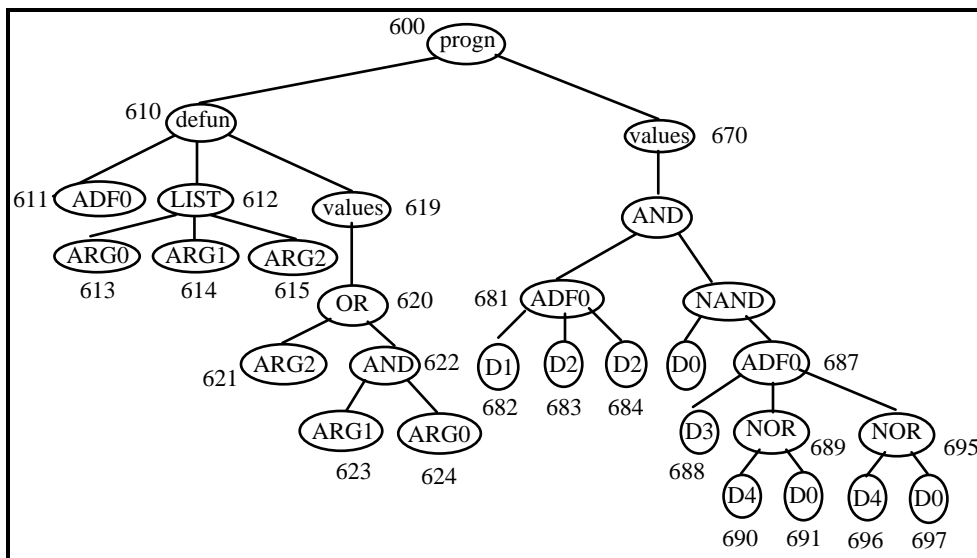
ARCHITECTURE-ALTERING OPERATIONS

PROGRAM WITH ARGUMENT MAP OF {2, 2} CREATED USING THE OPERATION OF BRANCH DUPLICATION



ARCHITECTURE-ALTERING OPERATIONS

PROGRAM WITH ARGUMENT MAP OF {3} CREATED USING THE OPERATION OF ARGUMENT DUPLICATION



ARCHITECTURE-ALTERING OPERATIONS

SPECIALIZATION – REFINEMENT – CASE SPLITTING

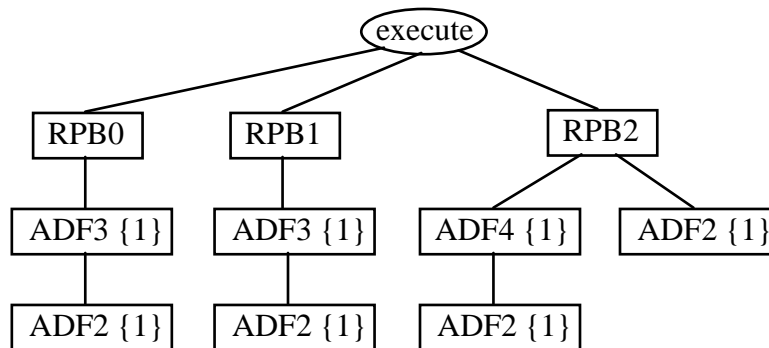
- **Branch duplication**
- **Argument duplication**
- **Branch creation**
- **Argument creation**

GENERALIZATION

- **Branch deletion**
- **Argument deletion**

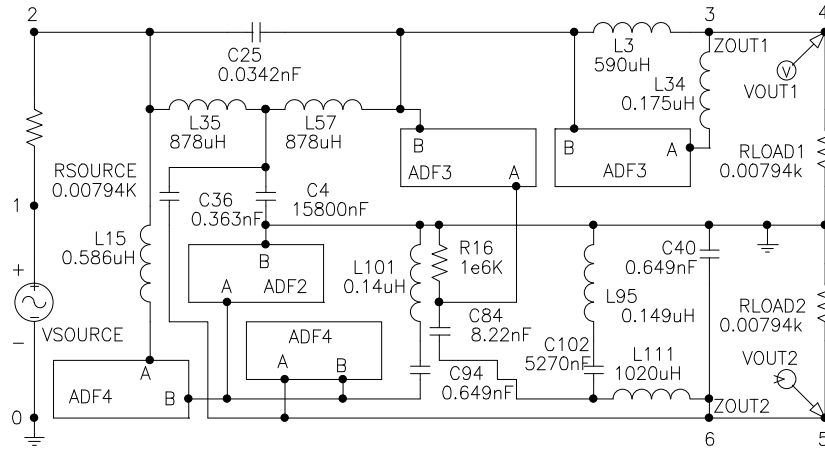
EMERGENCE OF A PARAMETERIZED ARGUMENT IN A CIRCUIT SUBSTRUCTURE

HIERARCHY OF BRANCHES FOR THE BEST-OF-RUN CIRCUIT- FROM GENERATION 158



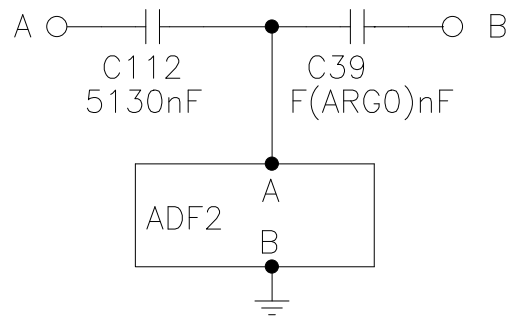
PASSING A PARAMETER TO A SUBSTRUCTURE

BEST-OF-RUN CIRCUIT FROM GENERATION 150



**THREE-PORTED AUTOMATICALLY
DEFINED FUNCTION ADF3 OF THE
BEST-OF-RUN CIRCUIT FROM
GENERATION 158**

**ADF3 CONTAINS CAPACITOR C39
PARAMETERIZED BY DUMMY
VARIABLE ARG0**



THE FIRST RESULT-PRODUCING BRANCH, RPB0, CALLING ADF3

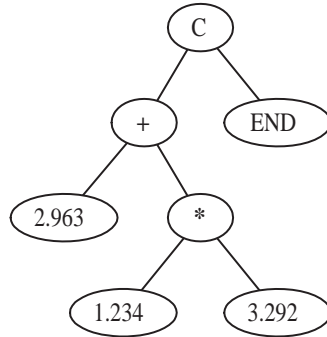
```
(PARALLEL0 (L (+ (- 1.883196E-01 (- -9.095883E-02 5.724576E-01)) (- 9.737455E-01 -9.452780E-01)) (FLIP END)) (SERIES (C (+ (+ -6.668774E-01 -8.770285E-01) 4.587758E-02) (NOP END)) (SERIES END END (PARALLEL1 END END END END)) (FLIP (SAFE_CUT))) (PAIR_CONNECT_0 END END END) (PAIR_CONNECT_0 (L (+ -7.220122E-01 4.896697E-01) END) (L (- -7.195599E-01 3.651142E-02) (SERIES (C (+ -5.111248E-01 (- (- -6.137950E-01 -5.111248E-01) (- 1.883196E-01 (- -9.095883E-02 5.724576E-01)))) END) (SERIES END END (adf3 6.196514E-01)) (NOP END))) (NOP END)))
```

AUTOMATICALLY DEFINED FUNCTION ADF3

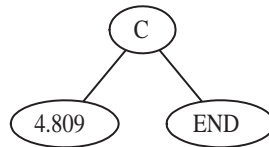
```
(C (+ (- (+ (+ (+ 5.630820E-01 (- 9.737455E-01 -9.452780E-01)) (+ ARG0 6.953752E-02)) (- (- 5.627716E-02 (+ 2.273517E-01 (+ 1.883196E-01 (+ 9.346950E-02 (+ -7.220122E-01 (+ 2.710414E-02 1.397491E-02)))))) (- (+ (- 2.710414E-02 -2.807583E-01) (+ -6.137950E-01 -8.554120E-01)) (- -8.770285E-01 (- -4.049602E-01 -2.192044E-02)))) (+ (+ 1.883196E-01 (+ (+ (+ (+ 9.346950E-02 (+ -7.220122E-01 (+ 2.710414E-02 1.397491E-02))) (- 4.587758E-02 -2.340137E-01)) 3.226026E-01) (+ -7.220122E-01 (- -9.131658E-01 6.595502E-01)))) 3.660116E-01) 9.496355E-01) (THREE_GROUND_0 (C (+ (- (+ (+ (+ 5.630820E-01 (- 9.737455E-01 -9.452780E-01)) (+ (- (- -7.195599E-01 3.651142E-02) -9.761651E-01) (- (+ (- (- -7.195599E-01 3.651142E-02) -9.761651E-01) 6.953752E-02) 3.651142E-02))) (- (- 5.627716E-02 (- 1.883196E-01 (- -9.095883E-02 5.724576E-01))) (- (+ (- 2.710414E-02 -2.807583E-01) (+ -6.137950E-01 (+ ARG0 6.953752E-02)) (- -8.770285E-01 (- -4.049602E-01 -2.192044E-02)))) (+ (+ 1.883196E-01 -7.195599E-01) 3.660116E-01) 9.496355E-01) (NOP (FLIP (PAIR_CONNECT_0 END END END))) (FLIP (SERIES (FLIP (FLIP (FLIP END))) (C (- (+ 6.238477E-01 6.196514E-01) (+ (+ (- (- 4.037348E-01 4.343444E-01) (+ -7.788187E-01 (+ (+ (- -8.786904E-01 1.397491E-02) (- -6.137950E-01 (- (+ (- 2.710414E-02 -2.807583E-01) (+ -6.137950E-01 -8.554120E-01)) (- -8.770285E-01 (- -4.049602E-01 -2.192044E-02)))) (+ (+ 7.215142E-03 1.883196E-01) (+ 7.733750E-01 4.343444E-01)))))) (- (- -9.389297E-01 5.630820E-01) (+ -5.840433E-02 3.568947E-01)) -8.554120E-01)) (NOP END)) END)) (FLIP (adf2 9.737455E-01)))
```

VALUE-SETTING SUBTREES—3 WAYS

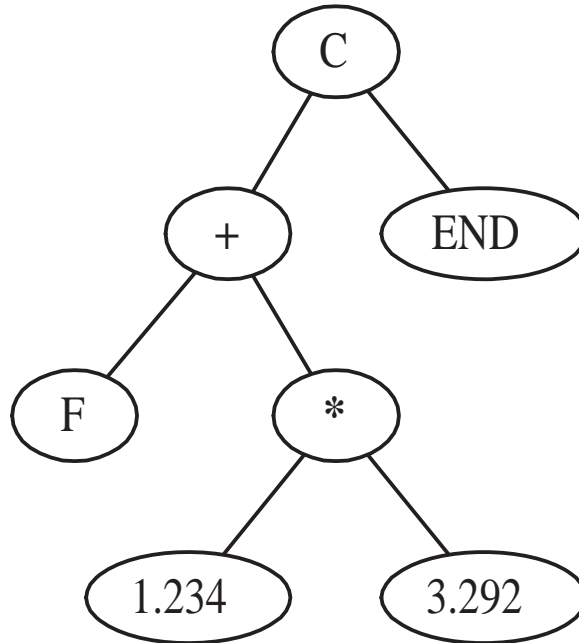
ARITHMETIC-PERFORMING SUBTREE



SINGLE PERTURBABLE CONSTANT



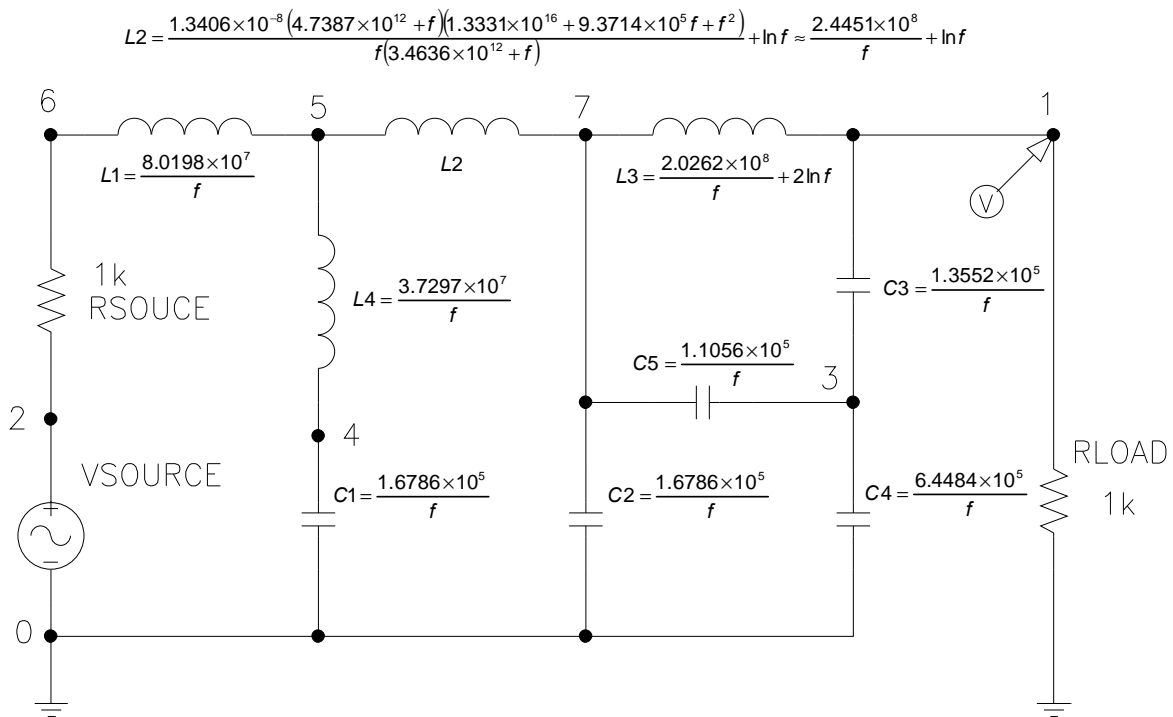
FREE VARIABLE



PARAMETERIZED TOPOLOGY FOR "GENERALIZED" LOWPASS FILTER

VARIABLE CUTOFF LOWPASS FILTER

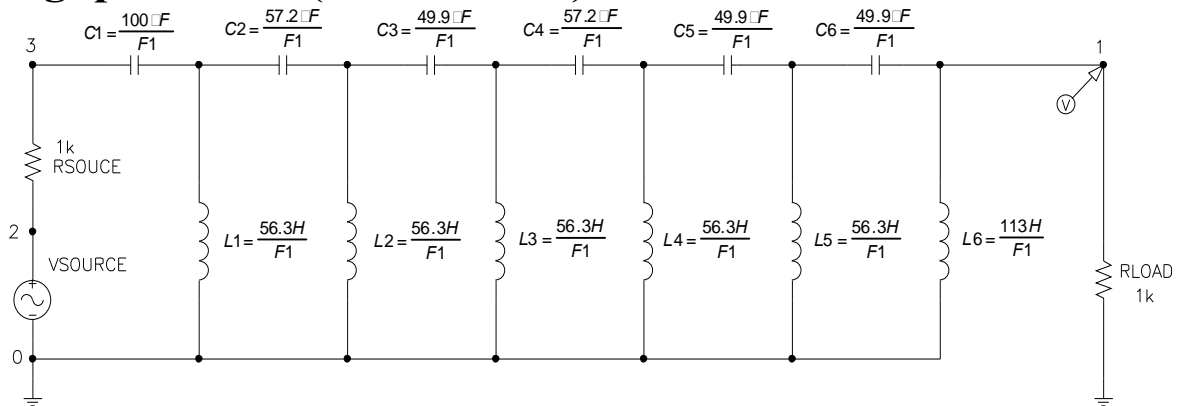
- Want lowpass filter whose passband ends at frequencies $f = 1,000, 1,780, 3,160, 5,620, 10,000, 17,800, 31,600, 56,200, 100,000$ Hz



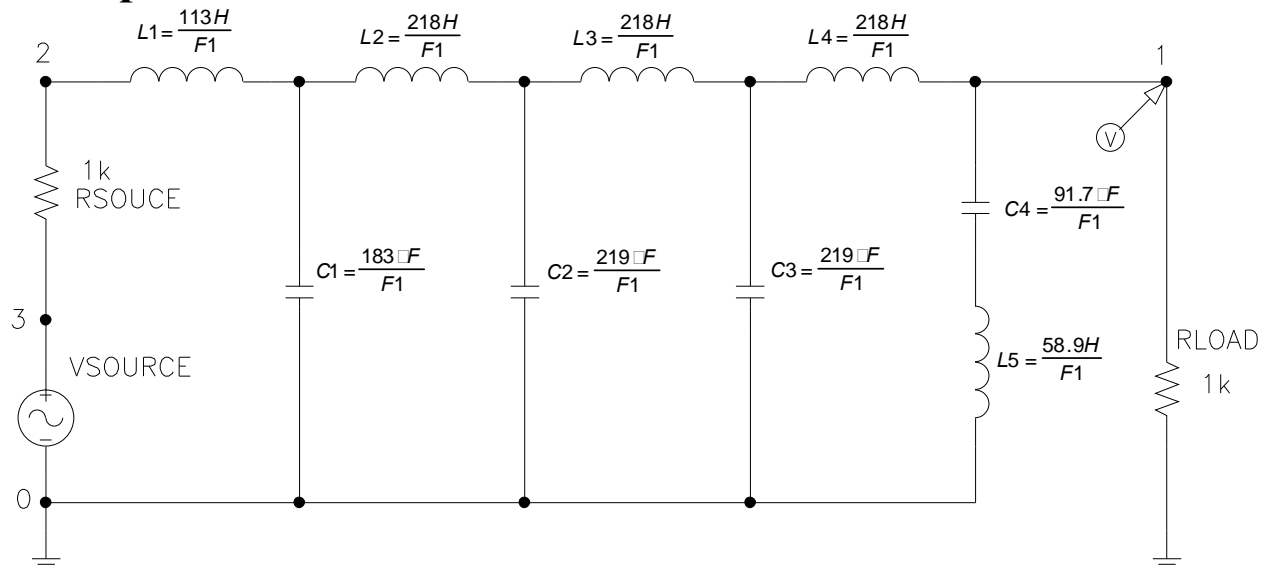
PARAMETERIZED TOPOLOGY USING CONDITIONAL DEVELOPMENTAL OPERATORS (GENETIC SWITCH)

VARIABLE-CUTOFF LOWPASS/HIGHPASS FILTER CIRCUIT

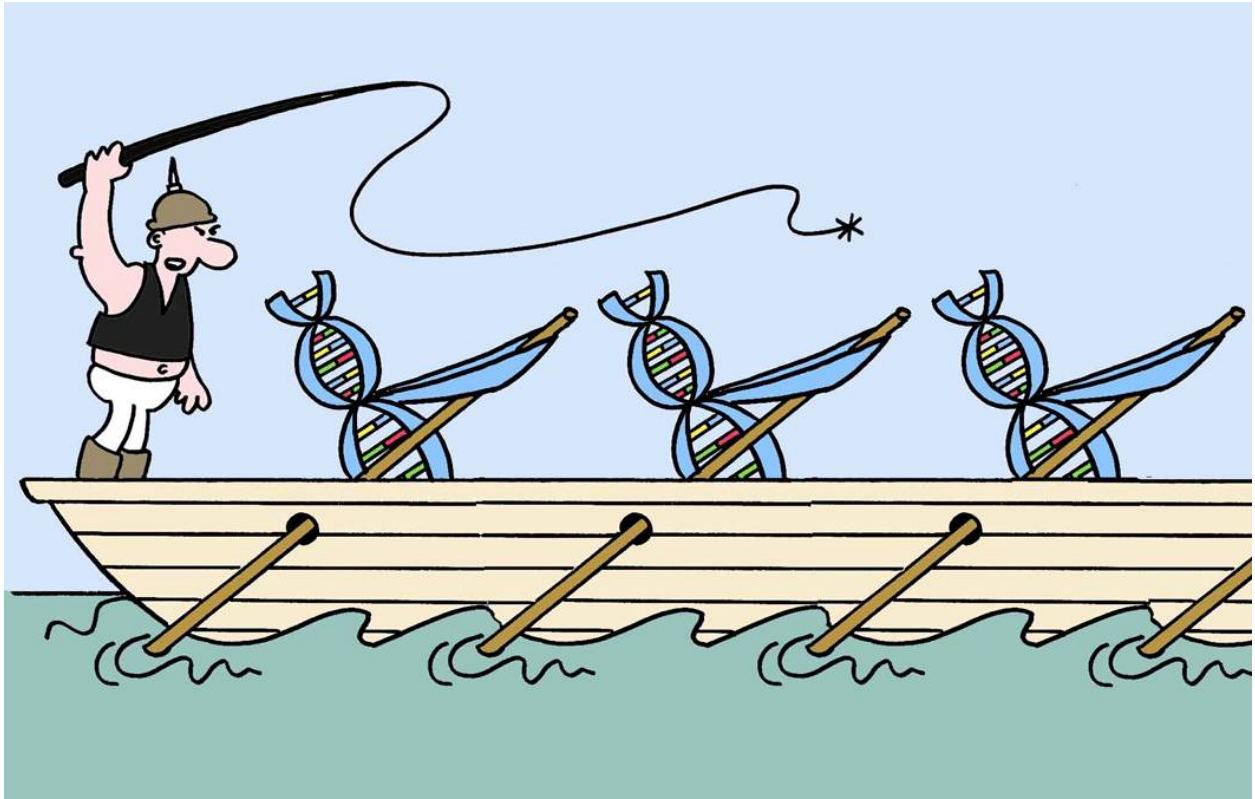
- Best-of-run circuit from generation 93 when inputs call for a highpass filter (i.e., $F1 > F2$).



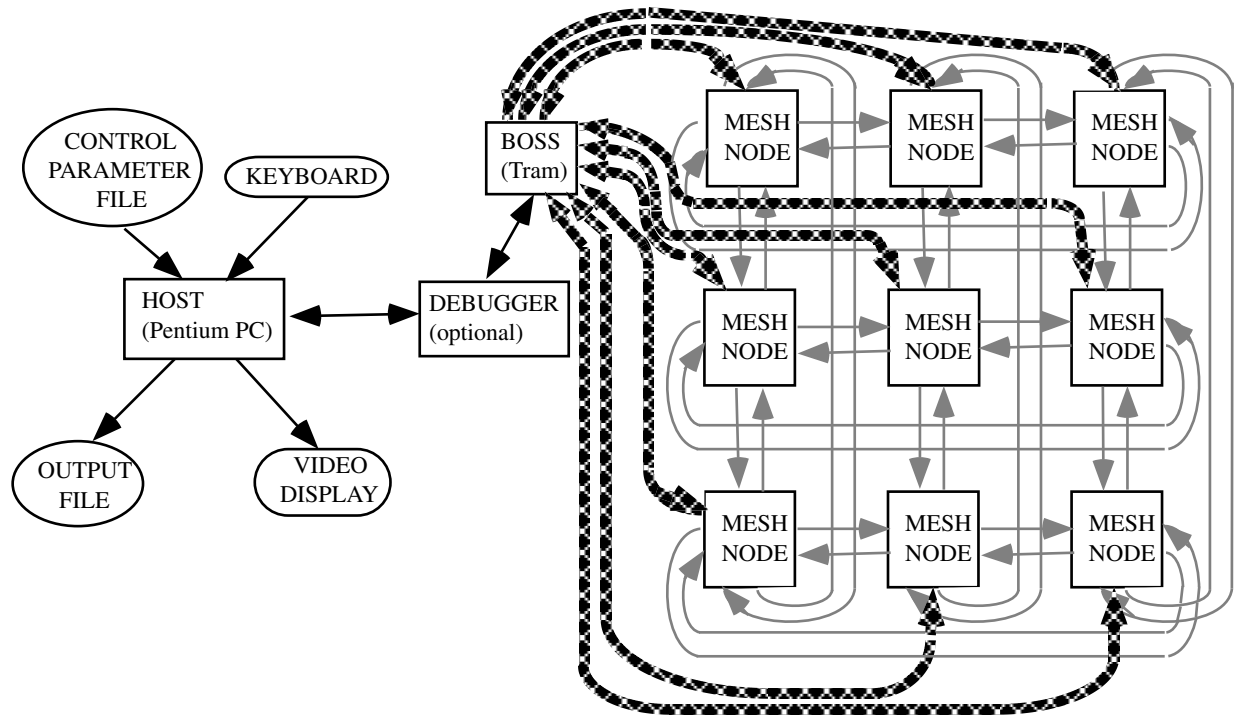
- Best-of-run circuit from generation 93 when inputs call for a lowpass filter.



PARALLELIZATION



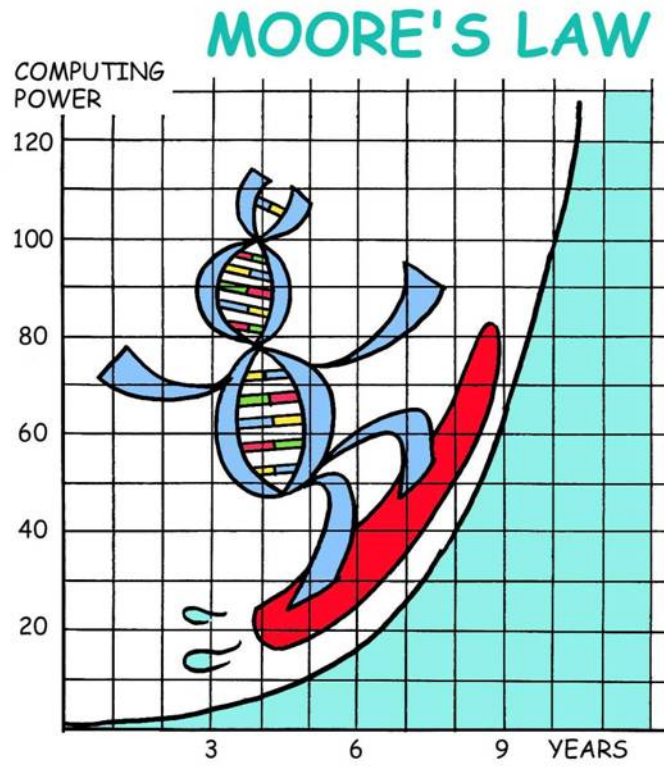
PARALLELIZATION BY SUBPOPULATIONS ("ISLAND" OR "DEME" MODEL OR "DISTRIBUTED GENETIC ALGORITHM")



- Like Hormel, Get Everything Out of the Pig, Including the Oink
- Keep on Trucking
- It Takes a Licking and Keeps on Ticking
- The Whole is Greater than the Sum of the Parts

PETA-OPS

- Human brain operates at 10^{12} neurons operating at 10^3 per second = 10^{15} ops per second
- 10^{15} ops = 1 peta-op = 1 bs (brain second)



GENETIC PROGRAMMING OVER 15- YEAR PERIOD 1987–2002

System	Period of usage	Petacycles (10^{15} cycles) per day for entire system	Speed-up over previous system	Speed-up over first system in this table	Human-competitive results
Serial Texas Instruments LISP machine	1987–1994	0.00216	1 (base)	1 (base)	0
64-node Transtech transputer parallel machine	1994–1997	0.02	9	9	2
64-node Parsytec parallel machine	1995–2000	0.44	22	204	12
70-node Alpha parallel machine	1999–2001	3.2	7.3	1,481	2
1,000-node Pentium II parallel machine	2000–2008	30.0	9.4	13,900	12

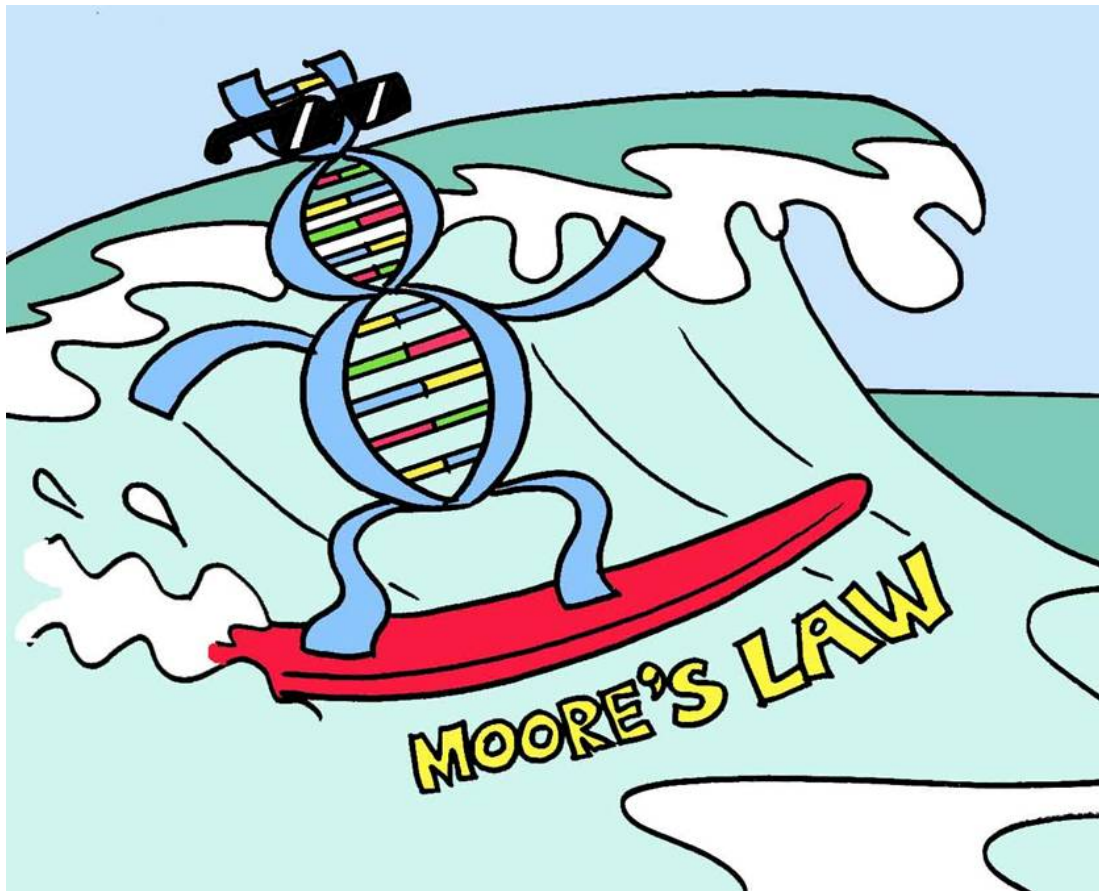
PROGRESSION OF RESULTS

System	Period	Speed-up	Qualitative nature of the results produced by genetic programming
Serial LISP machine	1987–1994	1 (base)	<ul style="list-style-type: none"> • Toy problems of the 1980s and early 1990s from the fields of artificial intelligence and machine learning
64-node Transtech 8-biy transputer	1994–1997	9	<ul style="list-style-type: none"> • Two human-competitive results involving one-dimensional discrete data (not patent-related)
64-node Parsytec parallel machine	1995–2000	22	<ul style="list-style-type: none"> • One human-competitive result involving two-dimensional discrete data • Numerous human-competitive results involving continuous signals analyzed in the frequency domain • Numerous human-competitive results involving 20th-century patented inventions
70-node Alpha parallel machine	1999–2001	7.3	<ul style="list-style-type: none"> • One human-competitive result involving continuous signals analyzed in the time domain • Circuit synthesis extended from topology and sizing to include routing and placement (layout)
1,000-node Pentium II parallel machine	2000–2002	9.4	<ul style="list-style-type: none"> • Numerous human-competitive results involving continuous signals analyzed in the time domain • Numerous general solutions to problems in the form of parameterized topologies • Six human-competitive results duplicating the functionality of 21st-century patented inventions
Long (4-week) runs of 1,000-node Pentium II parallel machine	2002	9.3	<ul style="list-style-type: none"> • Generation of two patentable new inventions

**PROGRESSION OF QUALITATIVELY
MORE SUBSTANTIAL RESULTS
PRODUCED BY GENETIC
PROGRAMMING IN RELATION TO FIVE
ORDER-OF-MAGNITUDE INCREASES IN
COMPUTATIONAL POWER**

- **toy problems**
- **human-competitive results not related to patented inventions**
- **20th-century patented inventions**
- **21st-century patented inventions**
- **patentable new inventions**

THE FUTURE



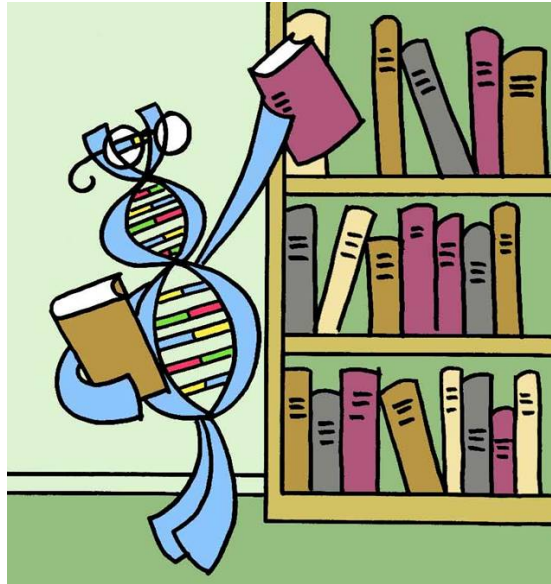
PROMISING GP APPLICATION AREAS

- Problem areas involving many variables that are interrelated in highly non-linear ways
- Inter-relationship of variables is not well understood
- A good approximate solution is satisfactory
 - design
 - control
 - classification and pattern recognition
 - data mining
 - system identification and forecasting
- Discovery of the size and shape of the solution (the “topology”) is a major part of the problem
- Areas where humans find it difficult to write programs
 - parallel computers
 - cellular automata
 - multi-agent strategies / distributed AI
 - FPGAs
 - reconfigurable analog arrays
 - reconfigurable antenna
- "black art" problems
 - synthesis of topology and sizing of analog circuits
 - synthesis of topology and tuning of controllers
 - quantum computing circuits
 - synthesis of designs for antennas
- Areas where you simply have no idea how to program a solution, but where the objective (fitness measure) is clear

CHARACTERISTICS SUGGESTING THE USE OF GENETIC PROGRAMMING

- **Problem areas where large computerized databases are accumulating and computerized techniques are needed to analyze the data**
- **problems where substructures are important**
 - **reusing substructures,**
 - **discovering the number of substructures,**
 - **discovering the nature of the hierarchical references among substructures,**
 - **passing parameters to a substructure,**
 - **discovering the type of substructures (e.g., subroutines, iterations, loops, recursions, or storage),**
 - **discovering the number of arguments possessed by a substructure,**
- **discovering a general solution in the form of a parameterized topology containing free variables**
- **maintaining syntactic validity and locality by means of a developmental process**

AUTHORED BOOKS ON GP



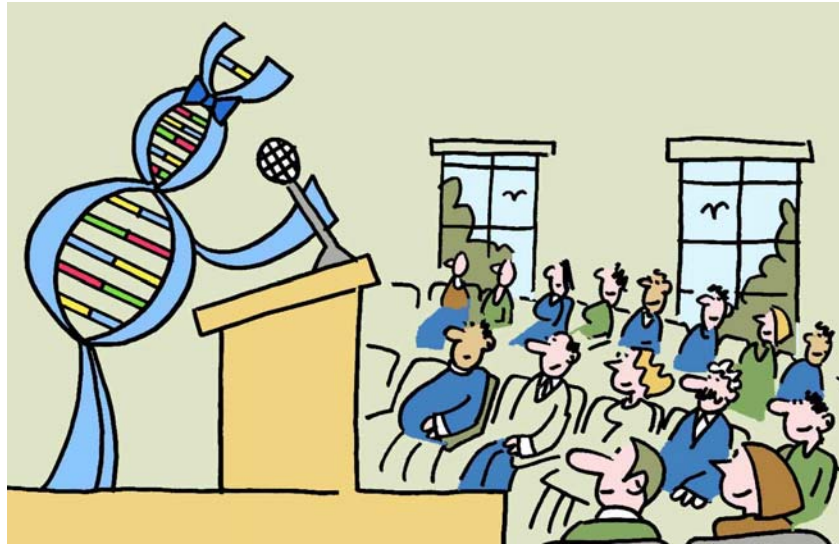
- Banzhaf, Wolfgang, Nordin, Peter, Keller, Robert E., and Francone, Frank D. 1998. *Genetic Programming - An Introduction*. San Francisco, CA: [Morgan Kaufman Publishers](#) and Heidelberg, Germany: dpunkt.verlag.
- Babovic, Vladan. 1996. *Emergence, Evolution, Intelligence: Hydroinformatics*. Rotterdam, The Netherlands: Balkema Publishers.
- Blickle, Tobias. 1997. *Theory of Evolutionary Algorithms and Application to System Synthesis*. TIK-Schriftenreihe Nr. 17. Zurich, Switzerland: [vdf Hochschul Verlag AG and der ETH Zurich](#). ISBN 3-7281-2433-8.
- Jacob, Christian. 1997. *Principia Evolvica: Simulierte Evolution mit Mathematica*. Heidelberg, Germany: dpunkt.verlag. In German. English translation forthcoming in 2000 from [Morgan Kaufman Publishers](#).
- Jacob, Christian. 2001. *Illustrating Evolutionary Computation with Mathematica*. San Francisco: Morgan Kaufmann.
- Iba, Hitoshi. 1996. *Genetic Programming*. Tokyo: Tokyo Denki University Press. In Japanese.
- Koza, John R. 1992. [Genetic Programming: On the Programming of Computers by Means of Natural Selection](#). Cambridge, MA: The MIT Press.
- Koza, John R. 1994. [Genetic Programming II: Automatic Discovery of Reusable Programs](#). Cambridge, MA: The MIT Press
- Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1999. [Genetic Programming III: Darwinian Invention and Problem Solving](#). San Francisco, CA: Morgan Kaufmann Publishers.
- Koza, John R., Keane, Martin A., Streeter, Matthew J., Mydlowec, William, Yu, Jessen, and Lanza, Guido. 2003. *Genetic Programming IV. Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

- Langdon, William B. 1998. *[Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!](#)* Amsterdam: Kluwer Academic Publishers.
- Langdon, William B. and Poli, Riccardo. 2002. *Foundations of Genetic Programming*. Berlin: Springer-Verlag.
- Nordin, Peter. 1997. *Evolutionary Program Induction of Binary Machine Code and its Application*. Munster, Germany: Krehl Verlag.
- O'Neill, Michael and Ryan, Conor. 2003. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Boston: Kluwer Academic Publishers.
- Ryan, Conor. 1999. *Automatic Re-engineering of Software Using Genetic Programming*. Amsterdam: Kluwer Academic Publishers.
- Spector, Lee. 2004. *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Boston: Kluwer Academic Publishers.
- Wong, Man Leung and Leung, Kwong Sak. 2000. *Data Mining Using Grammar Based Genetic Programming and Applications*. Amsterdam: Kluwer Academic Publishers.

MAIN POINTS OF GP-1,2,3,4 BOOKS

Book	Main Points
1992	<ul style="list-style-type: none"> • Virtually all problems in artificial intelligence, machine learning, adaptive systems, and automated learning can be recast as a search for a computer program. • Genetic programming provides a way to successfully conduct the search for a computer program in the space of computer programs.
1994	<ul style="list-style-type: none"> • Scalability is essential for solving non-trivial problems in artificial intelligence, machine learning, adaptive systems, and automated learning. • Scalability can be achieved by reuse. • Genetic programming provides a way to automatically discover and reuse subprograms in the course of automatically creating computer programs to solve problems.
1999	<ul style="list-style-type: none"> • Genetic programming possesses the attributes that can reasonably be expected of a system for automatically creating computer programs.
2003	<ul style="list-style-type: none"> • Genetic programming now routinely delivers high-return human-competitive machine intelligence. • Genetic programming is an automated invention machine. • Genetic programming can automatically create a general solution to a problem in the form of a parameterized topology. • Genetic programming has delivered a progression of qualitatively more substantial results in synchrony with five approximately order-of-magnitude increases in the expenditure of computer time.

VARIOUS CONFERENCES



ASPGP

Asian-Pacific Workshop on Genetic Programming

www.aspgp.org

GECCO (includes annual Genetic Programming conference)

Genetic and Evolutionary Computation Conference

www.SigEvo.org

EURO-GP

European Conference on Genetic Programming

evostar.na.icar.cnr.it/EuroGP/EuroGP.html

GPTP

Genetic Programming Theory and Practice

www.cscs.umich.edu/events/gptp2009/

3 EDITED *ADVANCES IN GENETIC PROGRAMMING* BOOKS

Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming 2*. Cambridge, MA: The MIT Press.

Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming*. Cambridge, MA: The MIT Press.

Spector, Lee, Langdon, William B., O'Reilly, Una-May, and Angeline, Peter (editors). 1999. *Advances in Genetic Programming 3*. Cambridge, MA: The MIT Press.

4 VIDEOTAPES ON GP

Koza, John R., and Rice, James P. 1992. [*Genetic Programming: The Movie*](#). Cambridge, MA: The MIT Press.

Koza, John R. 1994b. [*Genetic Programming II Videotape: The Next Generation*](#). Cambridge, MA: The MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, David, Keane, Martin A., and Brave, Scott. 1999. *Genetic Programming III Videotape: Human-Competitive Machine Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers.

Koza, John R., Keane, Martin A., Streeter, Matthew J., Mydlowec, William, Yu, Jessen, Lanza, Guido, and Fletcher, David. 2003. *Genetic Programming IV Video: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

WILLIAM LANGDON'S BIBLIOGRAPHY ON GENETIC PROGRAMMING

This bibliography is the most extensive in the field and contains over 5,000 papers (as of January 2003) by over 1,000 authors.

Visit

<http://www.cs.bham.ac.uk/~wbl/biblio/>

or

<http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html>

GENETIC PROGRAMMING AND EVOLVABLE MACHINES JOURNAL

www.springer.com/computer/artificial/journal/10710

GENETIC PROGRAMMING BOOK SERIES

<http://www.cs.ucl.ac.uk/staff/W.Langdon/gpdata/series.html>

GP MAILING LIST



To subscribe to the Genetic Programming e-mail list,

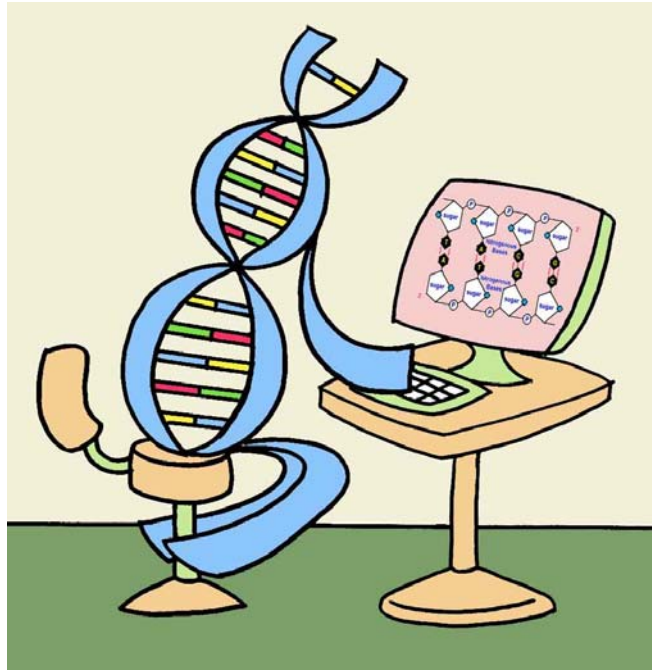
- **send e-mail message to:**

genetic_programming-subscribe@yahoogroups.com

- **visit the web page**

http://groups.yahoo.com/group/genetic_programming/

FOR ADDITIONAL INFORMATION



Visit

<http://www.genetic-programming.org>

and

<http://www.genetic-programming.com/coursemainpage.html>

for

- links computer code in various programming languages (including C, C++, Java, Mathematica, LISP)
- partial list of people active in genetic programming
- list of known completed PhD theses on GP
- list of students known to be working on PhD theses on GP
- information for instructors of university courses on genetic algorithms and genetic programming

GENETIC PROGRAMMING

