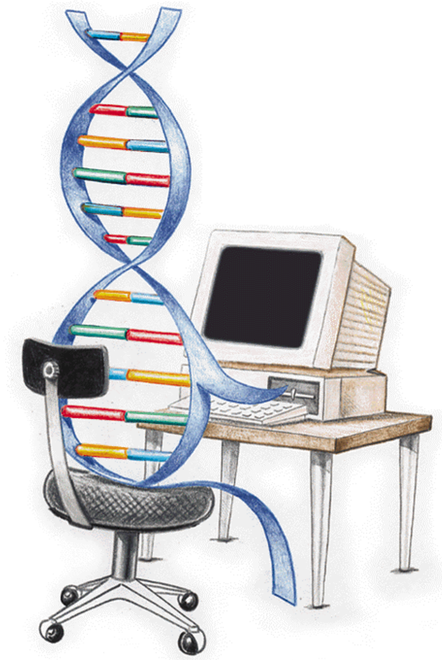
AUTOMATED INVENTION BY MEANS OF GENETIC PROGRAMMING



AAAI-2004 TUTORIAL—SAN JOSE
SUNDAY JULY 25, 2004—9AM

John R. Koza
Stanford University
koza@stanford.edu
http://smi-web.stanford.edu/people/koza/
http://www.genetic-programming.org

Lee Spector
Hampshire College
lspector@hampshire.edu
http://hampshire.edu/lspector

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

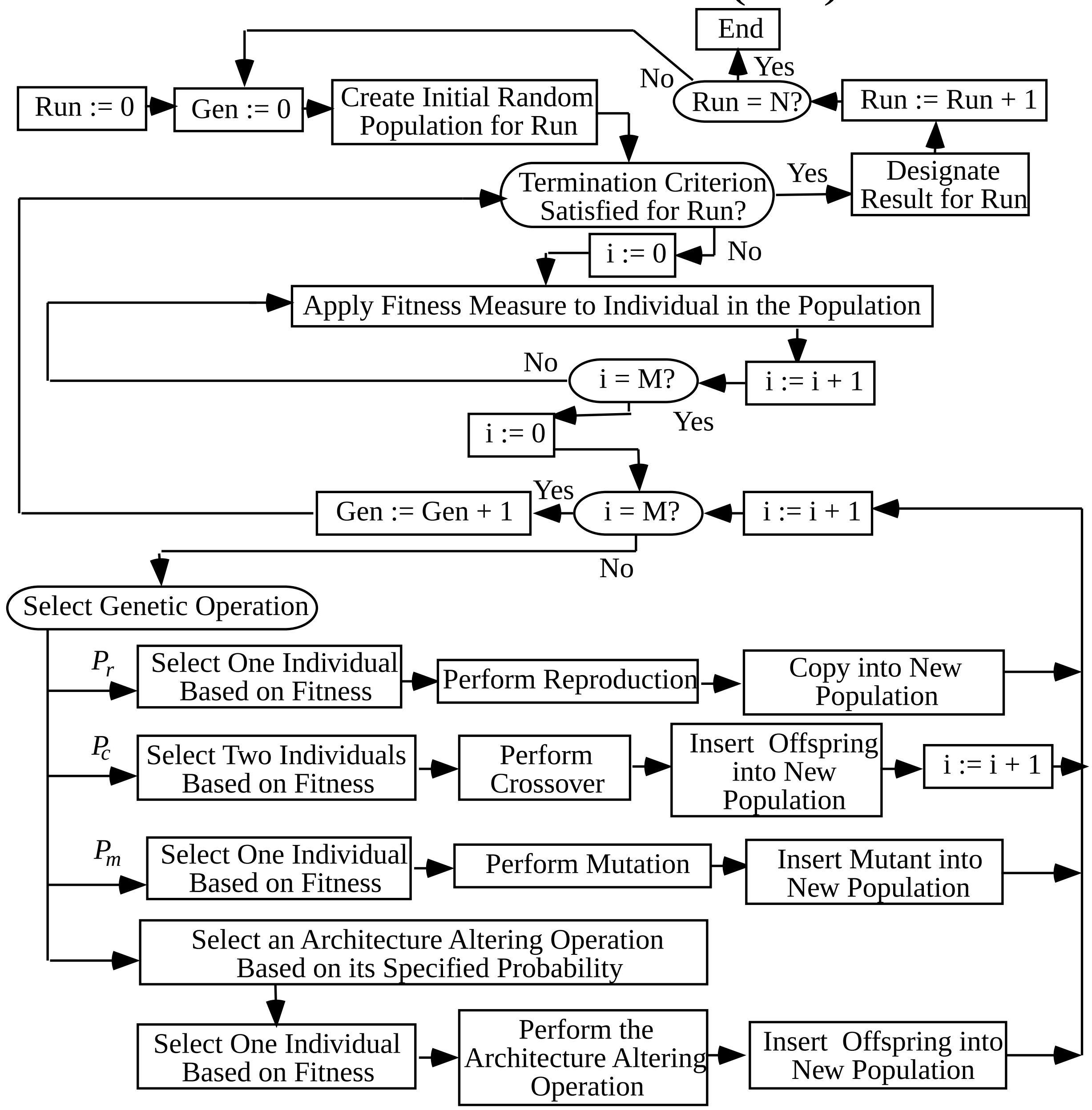
"The aim [is] ... to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence."

— Arthur Samuel (1983)

MAIN POINTS OF TUTORIAL

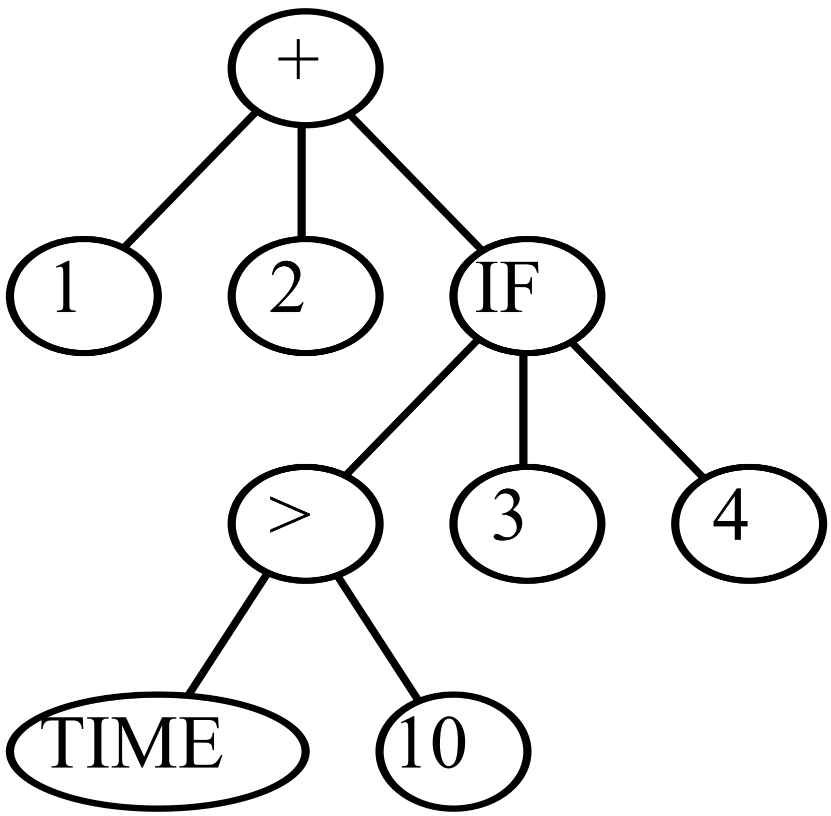
- **Genetic programming now routinely delivers high-return human-competitive machine intelligence**
- **Genetic programming is an automated invention machine**
- **Genetic programming has delivered a progression of qualitatively more substantial results in synchrony with five approximately order-of-magnitude increases in the expenditure of computer time**

FLOWCHART FOR GENETIC PROGRAMMING (GP)



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

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




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

- Creation of initial population (GIF)
- Reproduction operation
- Mutation operation (GIF)
- Crossover (recombination) operation (GIF)

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
  - population size
  - number of generations
  - minor parameters
- Determining the method for designating a result and the criterion for terminating a run

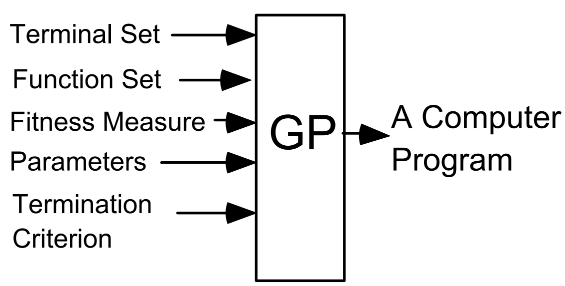
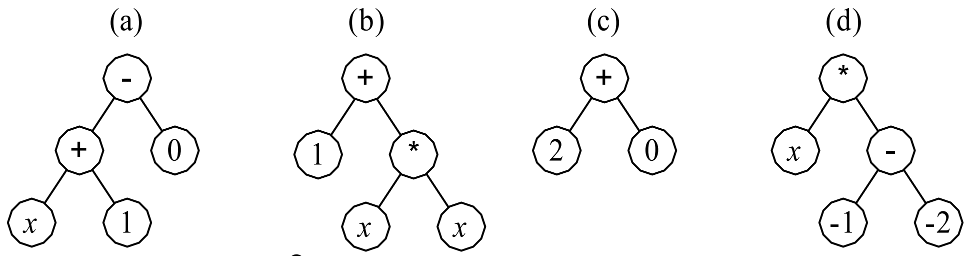


TABLEAU FOR SYMBOLIC REGRESSION OF QUADRATIC POLYNOMIAL $X^2 + X + 1$

	Objective:	Find a computer program with one input (independent variable x), whose output equals the value of the quadratic polynomial $x^2 + x + 1$ in range from -1 to +1.
1	Terminal set:	`T = {X}`
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

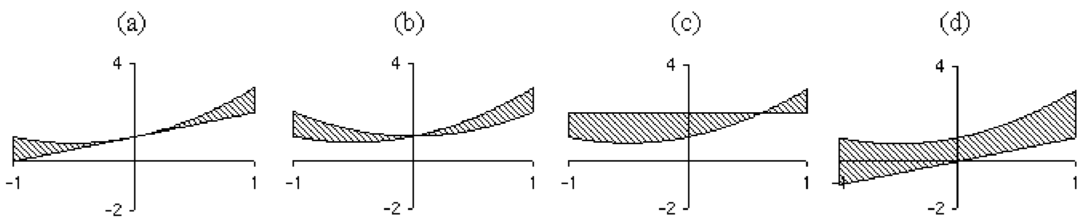
SYMBOLIC REGRESSION OF QUADRATIC POLYNOMIAL $X^2 + X + 1$

INITIAL POPULATION—GENERATION 0



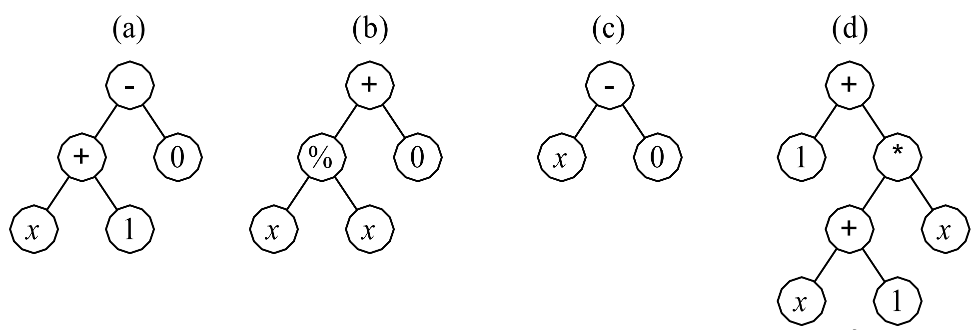
$X + 1$ $X^2 + 1$ 2 X

FITNESS



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

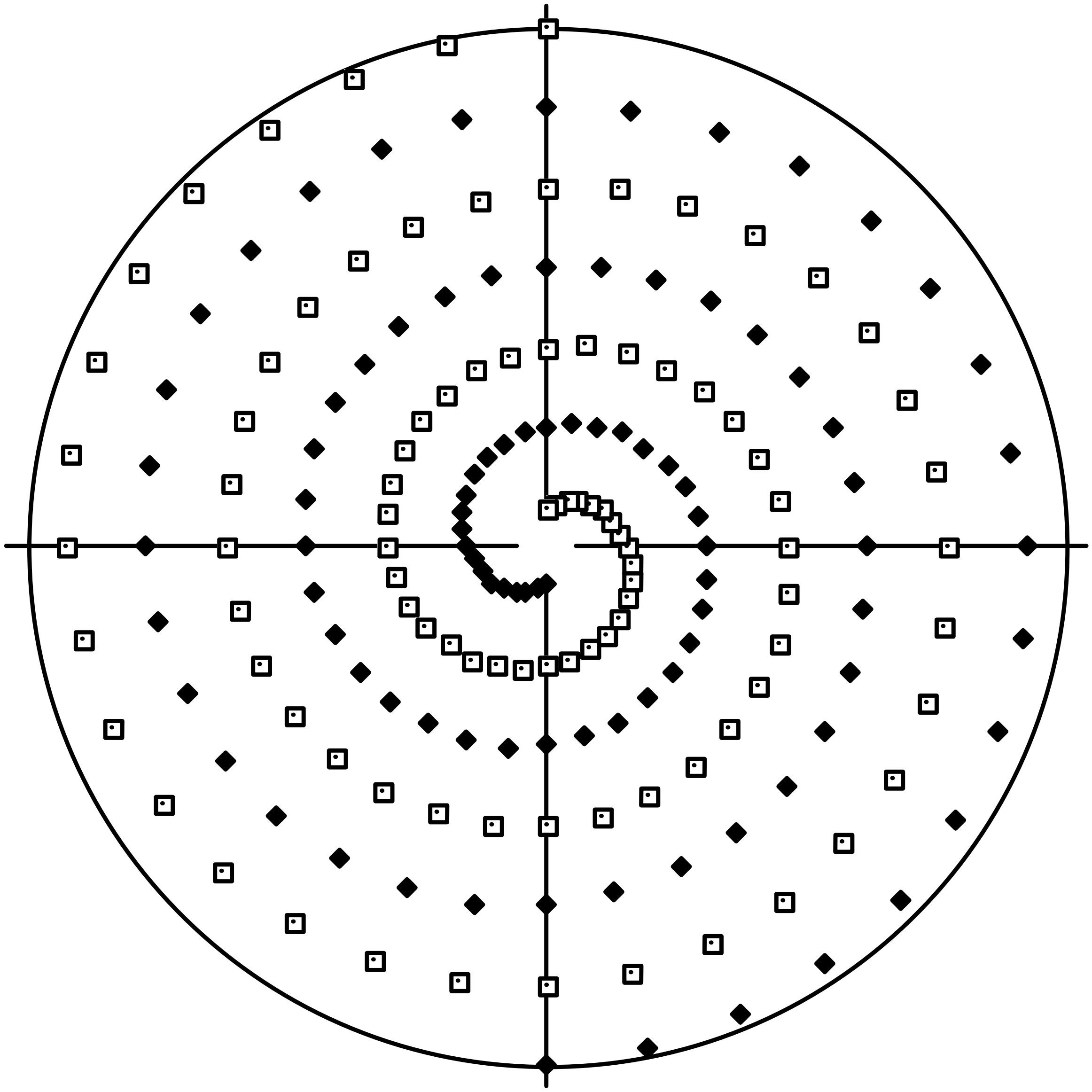
GENERATION 1



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

Copy of (a)	**Mutant of (c)** **—picking "2" as mutation point**	**First offspring of crossover of (a) and (b)** **—picking "+" of parent (a) and left-most "x" of parent (b) as crossover points**	**Second offspring of crossover of (a) and (b)** **—picking "+" of parent (a) and left-most "x" of parent (b) as crossover points**

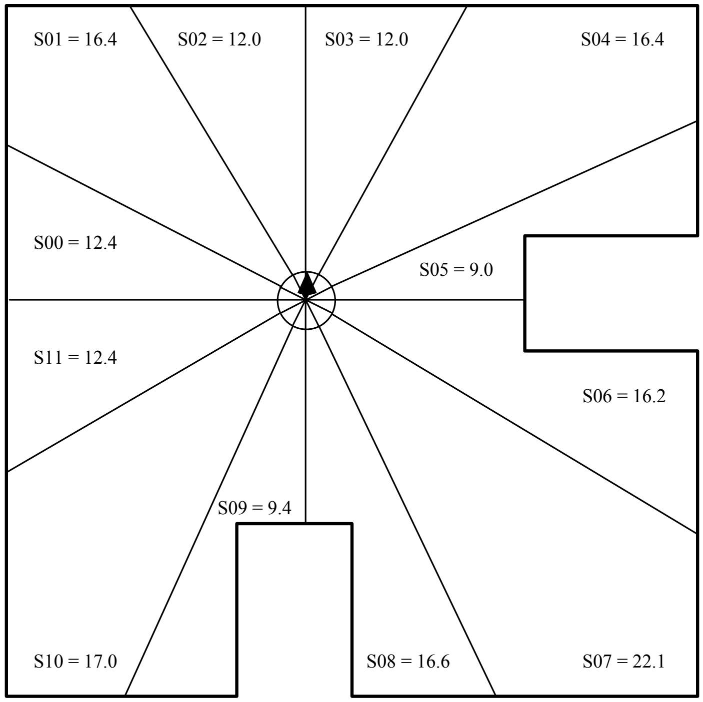
CLASSIFICATION PROBLEM
INTER-TWINED SPIRALS



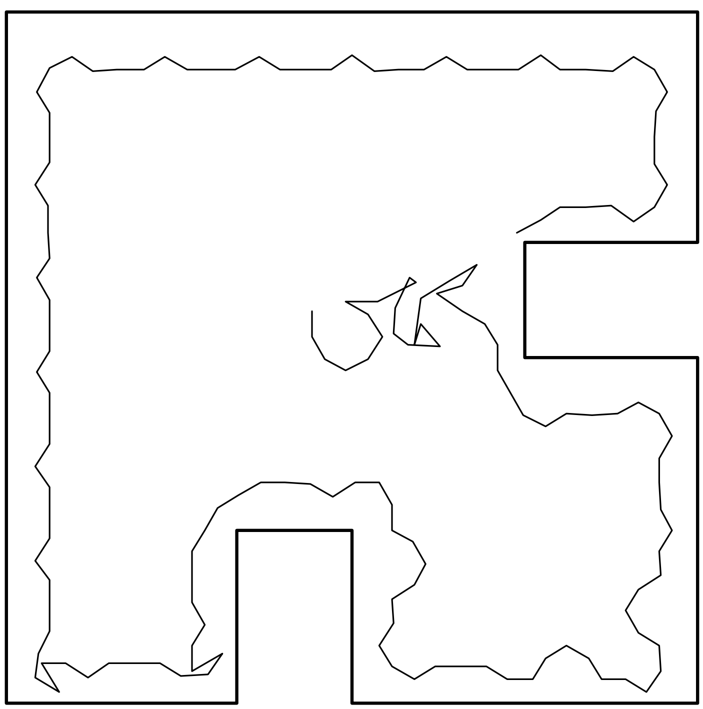
GP TABLEAU – INTERTWINED SPIRALS

Objective:	**Find a program to classify a given point in the *x*-*y* plane to the red or blue spiral.**
Terminal set:	**`X`, `Y`, $\Re$, where $\Re$ 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



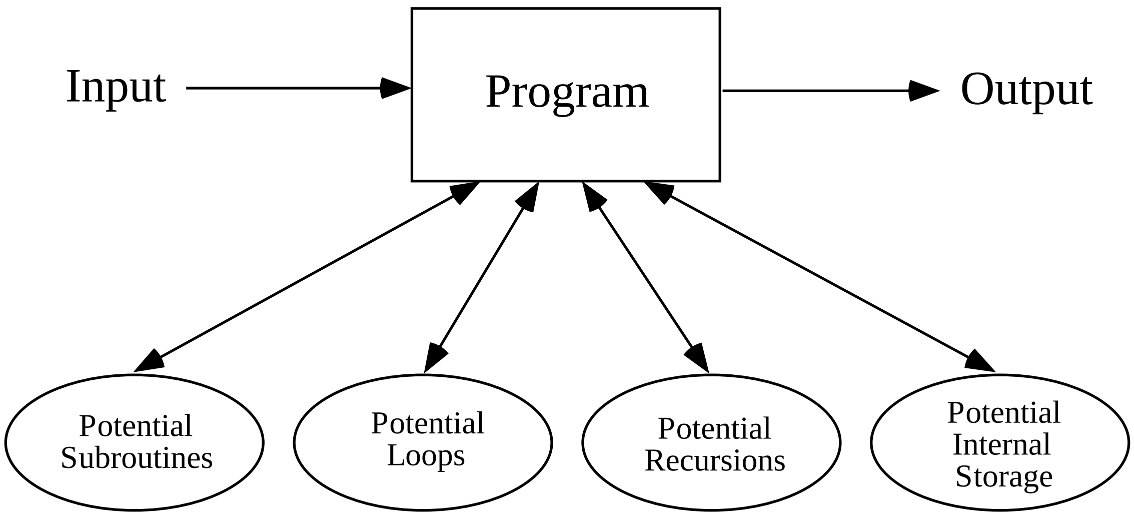
EVOLVED WALL-FOLLOWER



SOME OF THE PROBLEMS SOLVED IN *GENETIC PROGRAMMING* (KOZA 1992)

- Symbolic Regression
- Intertwined Spirals
- Wall Following
- Box Moving
- Truck Backer Upper
- Broom Balancing
- Artificial Ant
- 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

AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

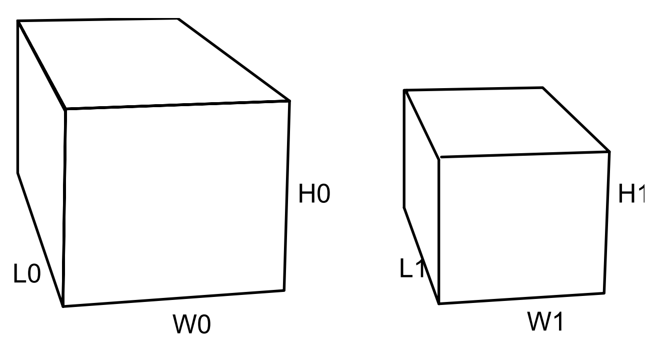


- **Subroutines provide one way to REUSE code — possibly with different instantiations of the dummy variables (formal parameters)**
- **Loops (and iterations) provide a 2nd way to REUSE code**
- **Recursion provide a 3rd way to REUSE code**
- **Memory provides a 4th way — to REUSE the results of executing code**

AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

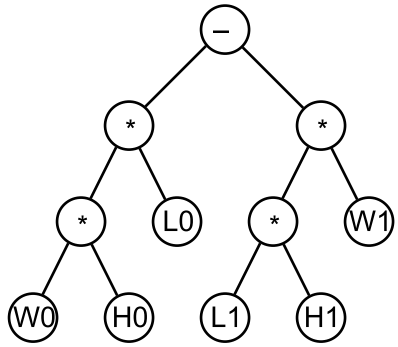
10 FITNESS-CASES SHOWING THE VALUE OF THE DEPENDENT VARIABLE, *D*, ASSOCIATED WITH THE VALUES OF THE SIX INDEPENDENT VARIABLES, L_0, W_0, H_0, L_1, W_1, H_1

Fitness case	L_0	W_0	H_0	L_1	W_1	H_1	**Dependent variable *D***
1	3	4	7	2	5	3	**54**
2	7	10	9	10	3	1	**600**
3	10	9	4	8	1	6	**312**
4	3	9	5	1	6	4	**111**
5	4	3	2	7	6	1	**–18**
6	3	3	1	9	5	4	**–171**
7	5	9	9	1	7	6	**363**
8	1	2	9	3	9	2	**–36**
9	2	6	8	2	6	10	**–24**
10	8	1	10	7	5	1	**45**



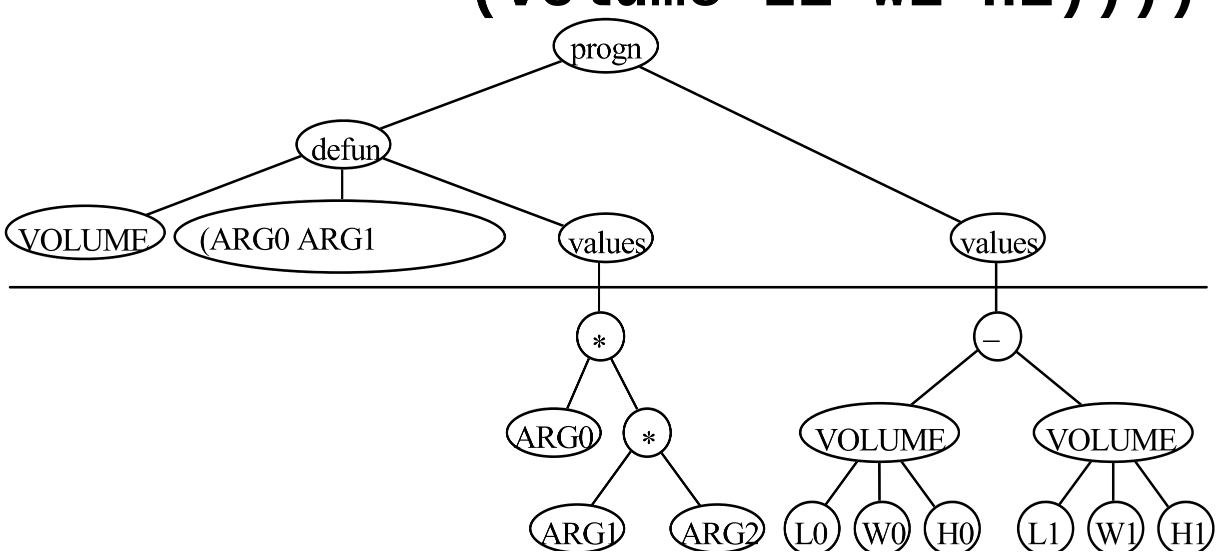
SOLUTION WITHOUT ADFs

```
(- (* (* W0 L0) H0)
   (* (* W1 L1) H1))
```



SOLUTION WITH ADFs

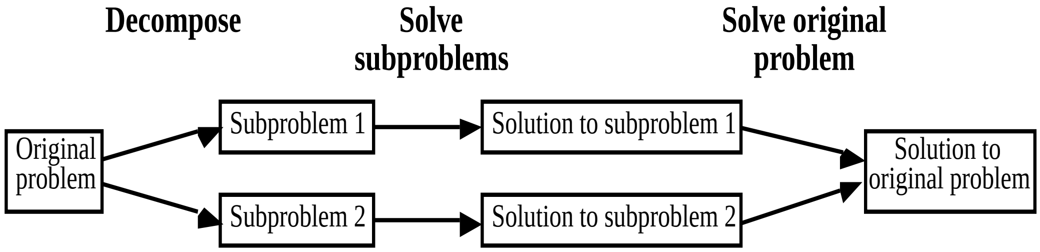
```
(progn
 (defun volume (arg0 arg1 arg2)
    (values
       (* arg0 (* arg1 arg2))))
(values  (- (volume L0 W0 H0)
            (volume L1 W1 H1))))
```



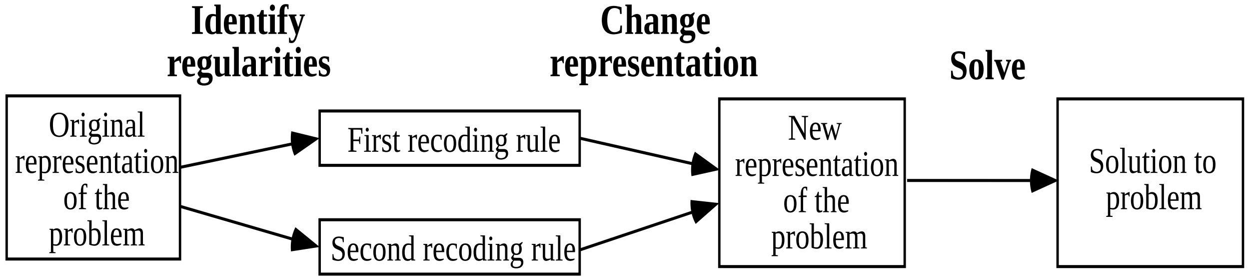
ADFs (SUBROUTINES)

Fitness case	L_0	W_0	H_0	L_1	W_1	H_1	V_0	V_1	D
1	3	4	7	2	5	3	**84**	**30**	**54**
2	7	10	9	10	3	1	**630**	**30**	**600**
3	10	9	4	8	1	6	**360**	**48**	**312**
4	3	9	5	1	6	4	**135**	**24**	**111**
5	4	3	2	7	6	1	**24**	**42**	**–18**
6	3	3	1	9	5	4	**9**	**180**	**–171**
7	5	9	9	1	7	6	**405**	**42**	**363**
8	1	2	9	3	9	2	**18**	**54**	**–36**
9	2	6	8	2	6	10	**96**	**120**	**–24**
10	8	1	10	7	5	1	**80**	**35**	**45**

TOP-DOWN VIEW



BOTTOM-UP VIEW



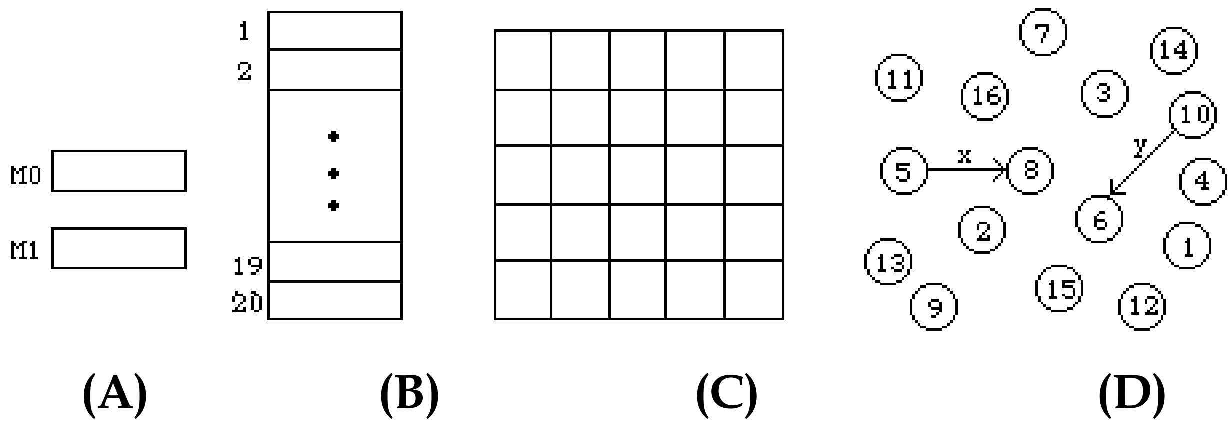
AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

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

- **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

MEMORY AND STORAGE



- (A) Settable (named) variables (*Genetic Programming*, Koza 1992) using setting (writing) functions `(SETM0 X)` and `(SETM1 Y)` and reading by means of terminals `M0` and `M1`.
- (B) Indexed memory similar to linear (vector) computer memory (Teller 1994) using `(READ K)` and `(WRITE X K)`
- (C) Matrix memory (Andre 1994)
- (D) Relational memory (Brave 1995, 1996)

LANGDON'S DATA STRUCTURES

- Stacks
- Queues
- Lists
- Rings

REUSE

AUTOMATICALLY DEFINED ITERATIONS (ADIs)

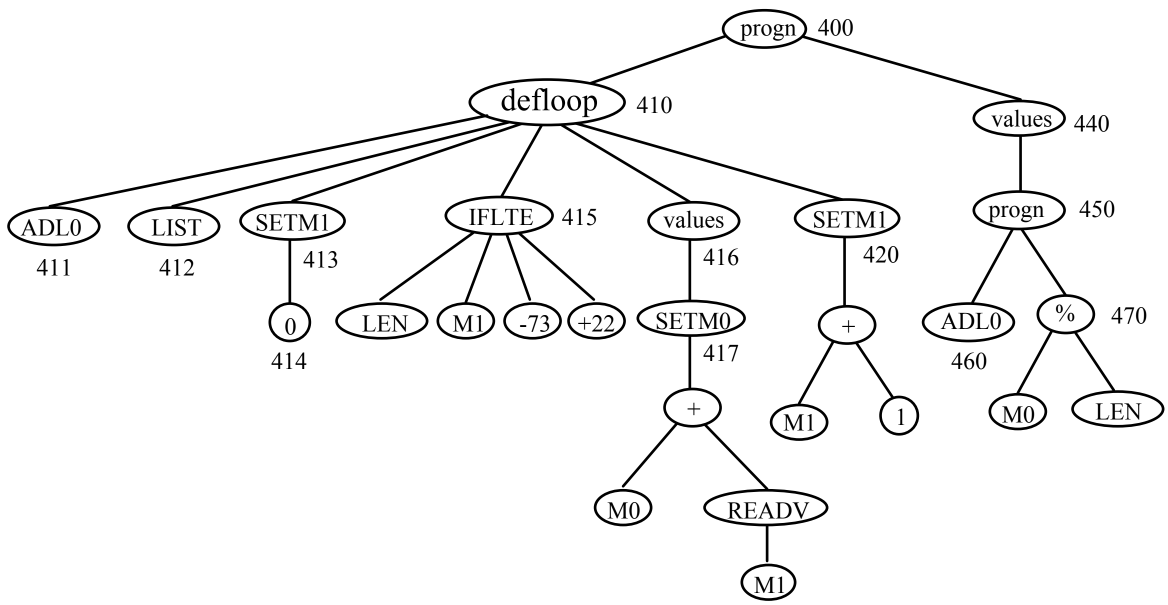
TRANSMEMBRANE SEGMENT IDENTIFICATION PROBLEM

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

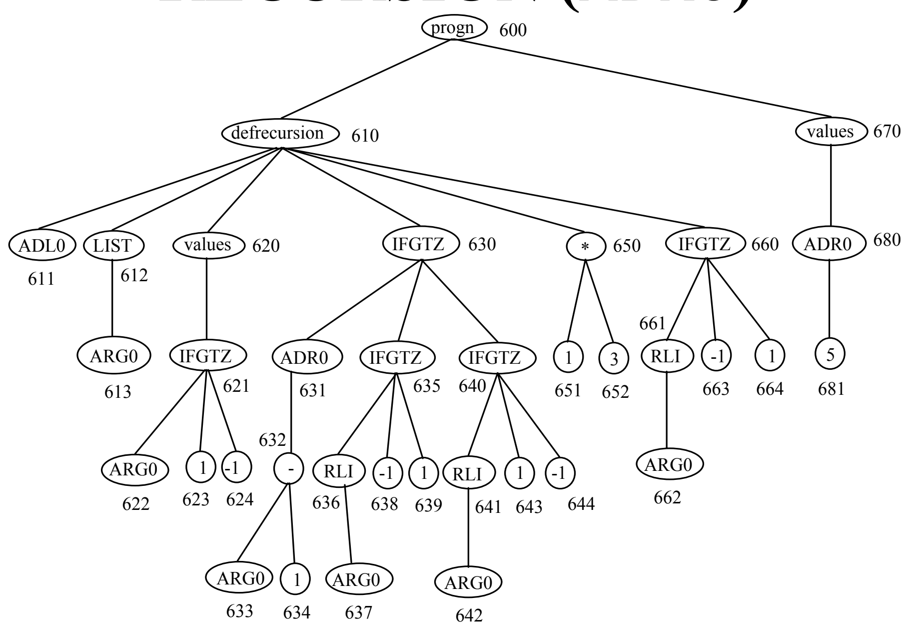
```
(progn

    (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))))))
```

REUSE

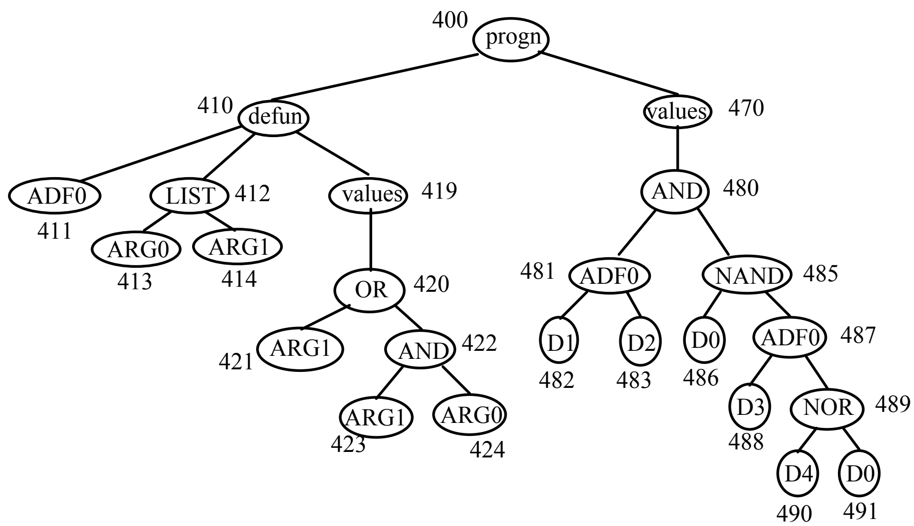
AUTOMATICALLY DEFINED LOOP (ADL0)

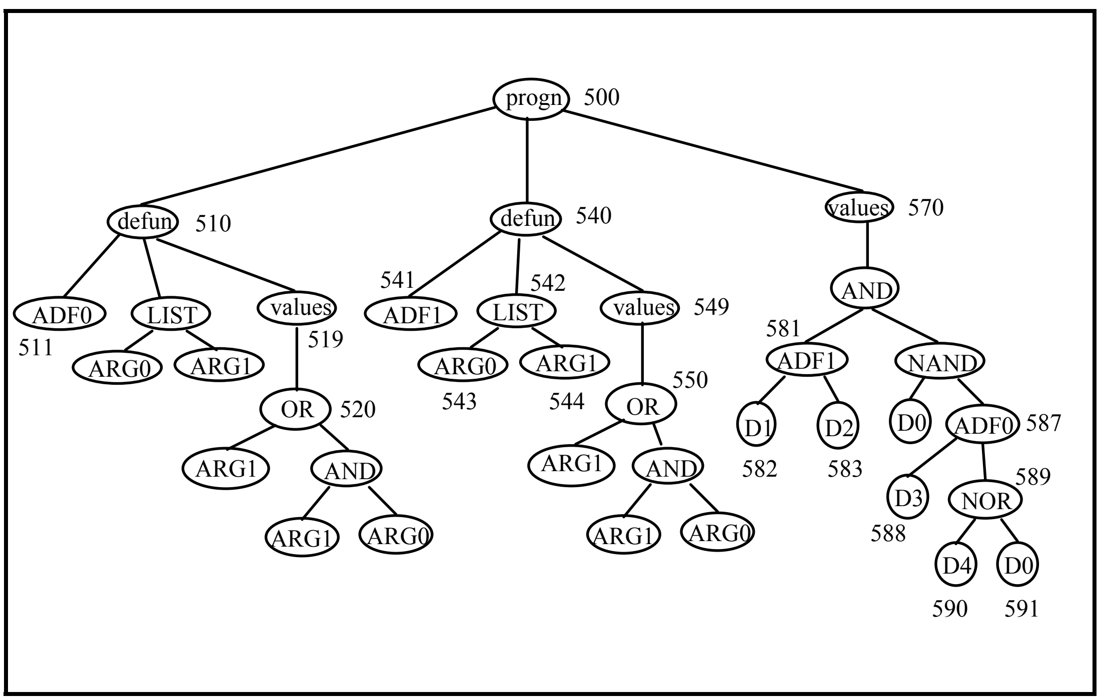


AUTOMATICALLY DEFINED RECURSION (ADR0)



ARCHITECTURE-ALTERING OPERATIONS





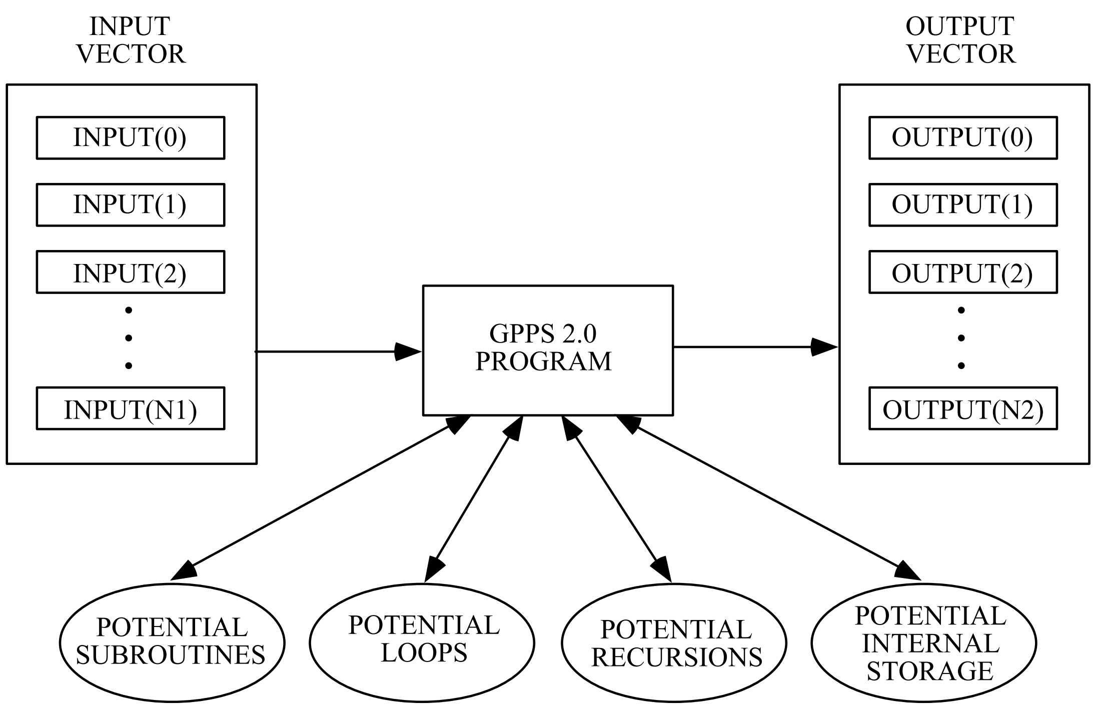
(GIFS)

16 ATTRIBUTES OF A SYSTEM FOR AUTOMATICALLY CREATING COMPUTER PROGRAMS

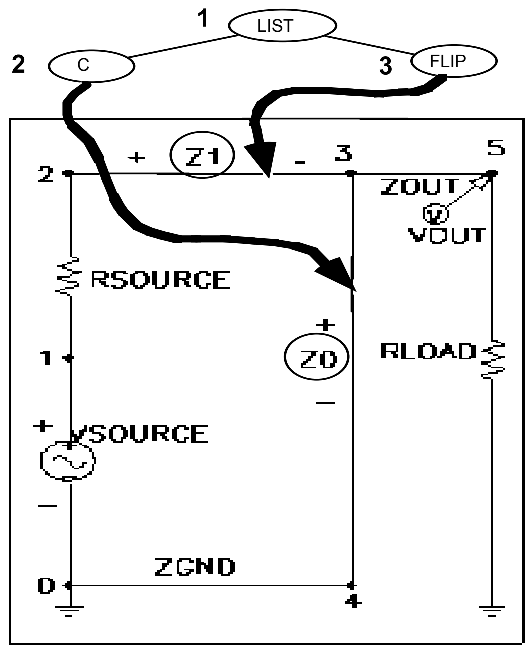
1 — Starts with "What needs to be done"
2 — Tells us "How to do it"
3 — Produces a computer program
4 — Automatic determination of program size
5 — Code reuse
6 — Parameterized reuse
7 — Internal storage
8 — Iterations, loops, and recursions
9 — Self-organization of hierarchies
10 — Automatic determination of program architecture
11 — Wide range of programming constructs
12 — Well-defined
13 — Problem-independent
14 — Wide applicability
15 — Scalable
16 — Competitive with human-produced results

ARCHITECTURE-ALTERING OPERATIONS

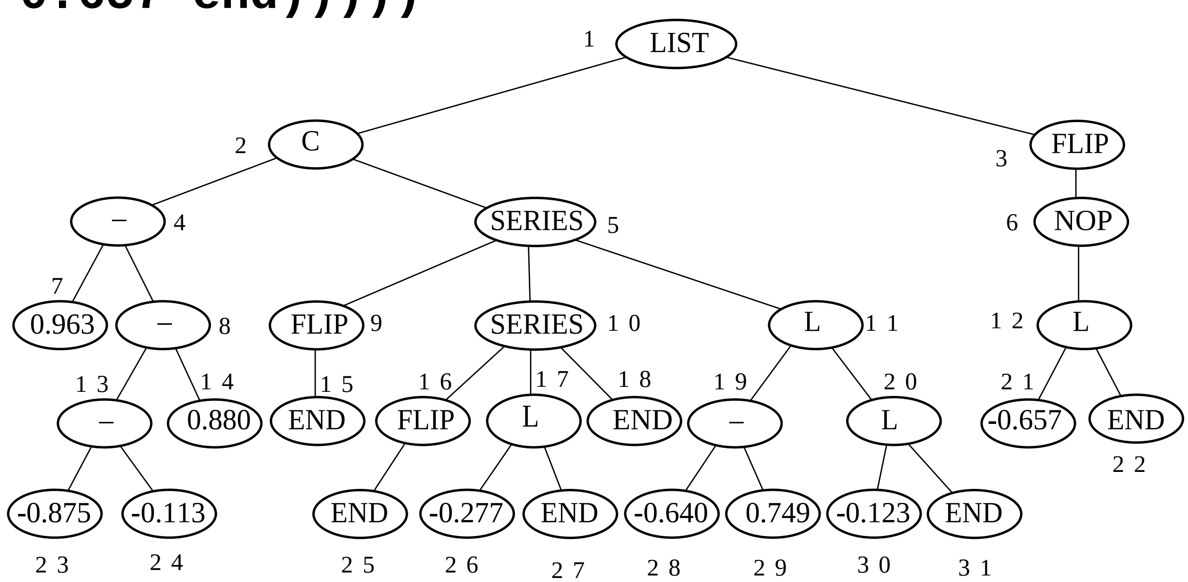
GENETIC PROGRAMMING PROBLEM SOLVER (GPPS) — VERSION 2.0



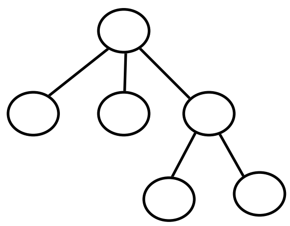
DEVELOPMENTAL GP



(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)))))

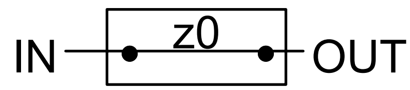


EVALUATION OF FITNESS OF A CIRCUIT



Program Tree

+



Embryonic Circuit

↓

Fully Designed Circuit (NetGraph)

↓

Circuit Netlist (ascii)

↓

Circuit Simulator (SPICE)

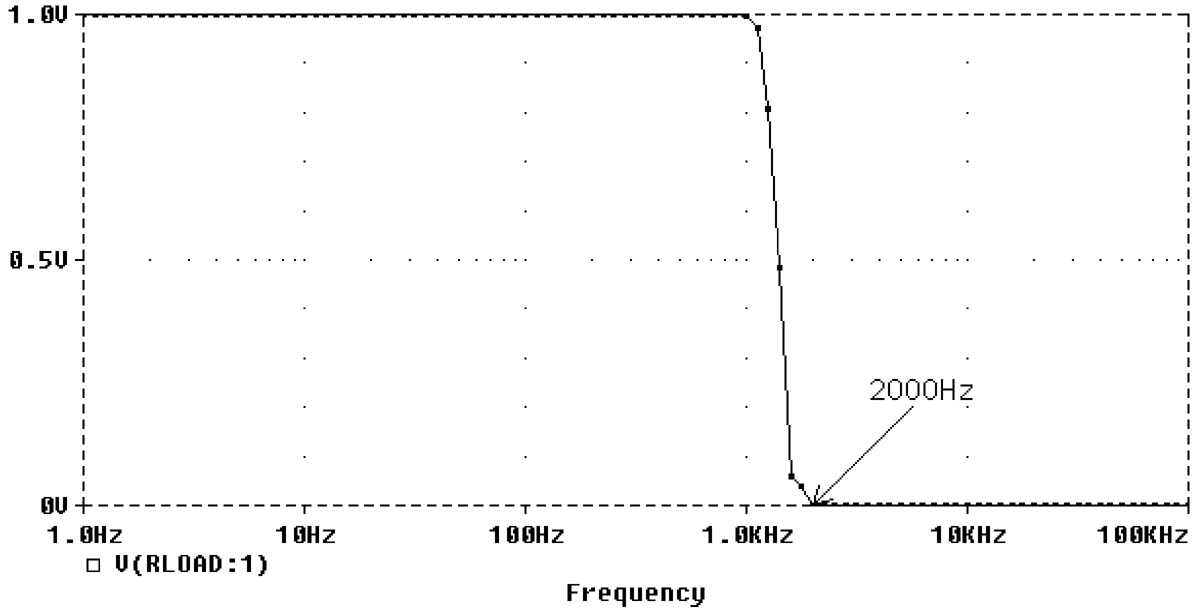
↓

Circuit Behavior (Output)

↓

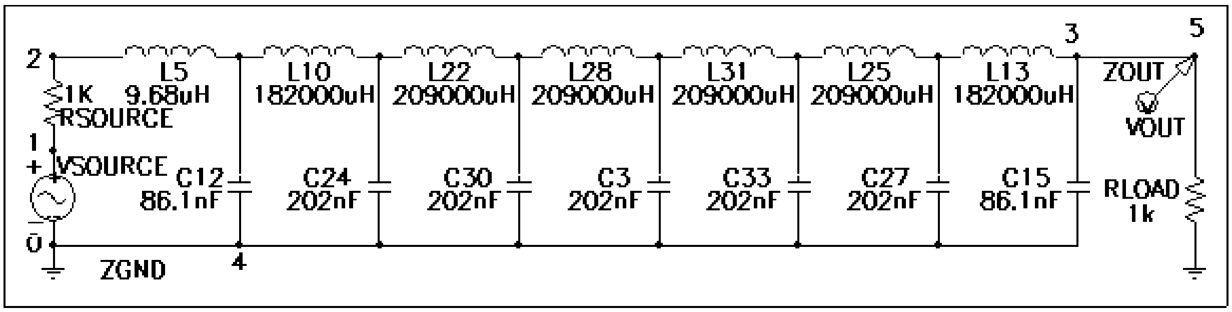
Fitness

BEHAVIOR OF A LOWPASS FILTER VIEWED IN THE FREQUENCY DOMAIN

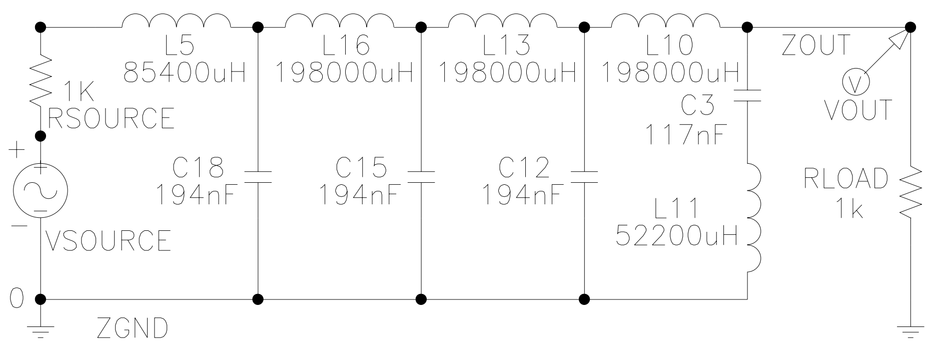


- **Examine circuit's behavior for each of 101 frequency values chosen over five decades of frequency (from 1 Hz to 100,000 Hz) with each decade divided into 20 parts (using a logarithmic scale). The fitness measure**
  - **does not penalize ideal values**
  - **slightly penalizes acceptable deviations**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

EVOLVED FILTER U. S. PATENT 1,227,113—GEORGE CAMPBELL—AT&T—1917

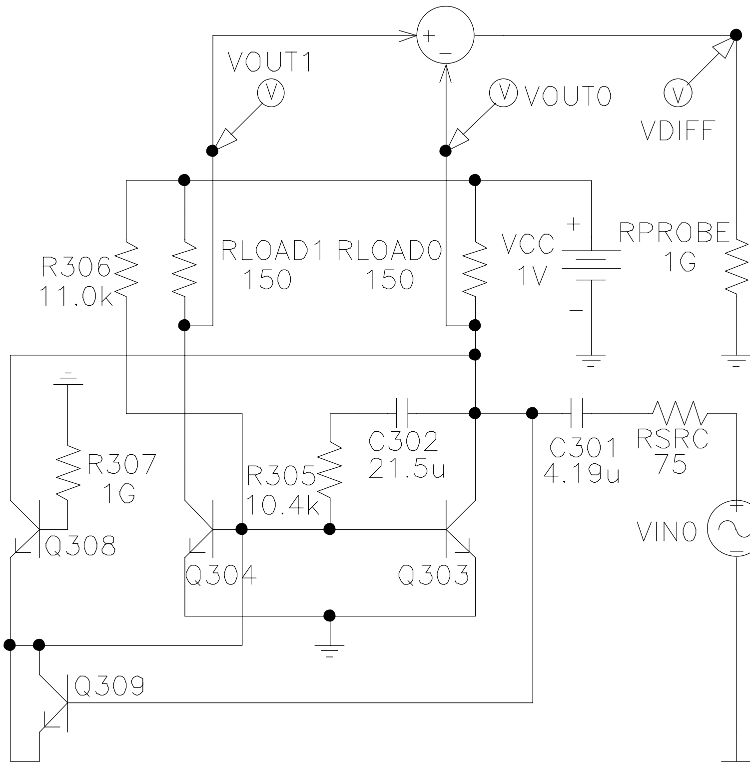


EVOLVED FILTER U. S. PATENT 1,538,964—OTTO ZOBEL—AT&T—1925



POST-2000 PATENTED INVENTIONS

LOW-VOLTAGE BALUN CIRCUIT



21 PREVIOUSLY PATENTED INVENTIONS REINVENTED BY GP

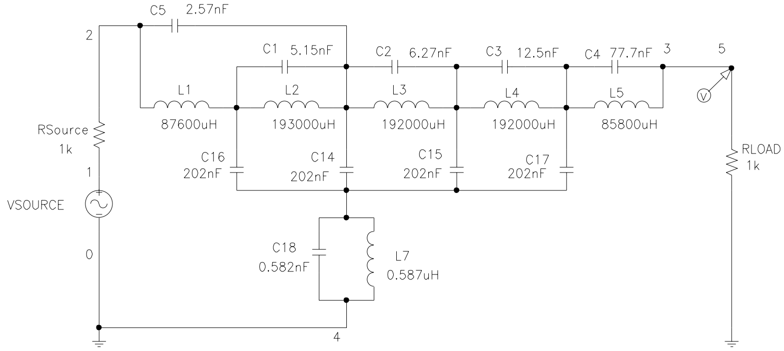
	Invention	Date	Inventor	Place	Patent
1	Darlington emitter-follower section	1953	Sidney Darlington	Bell Telephone Laboratories	2,663,806
2	Ladder filter	1917	George Campbell	American Telephone and Telegraph	1,227,113
3	Crossover filter	1925	Otto Julius Zobel	American Telephone and Telegraph	1,538,964
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

2 PATENTABLE INVENTIONS CREATED BY GENETIC PROGRAMMING

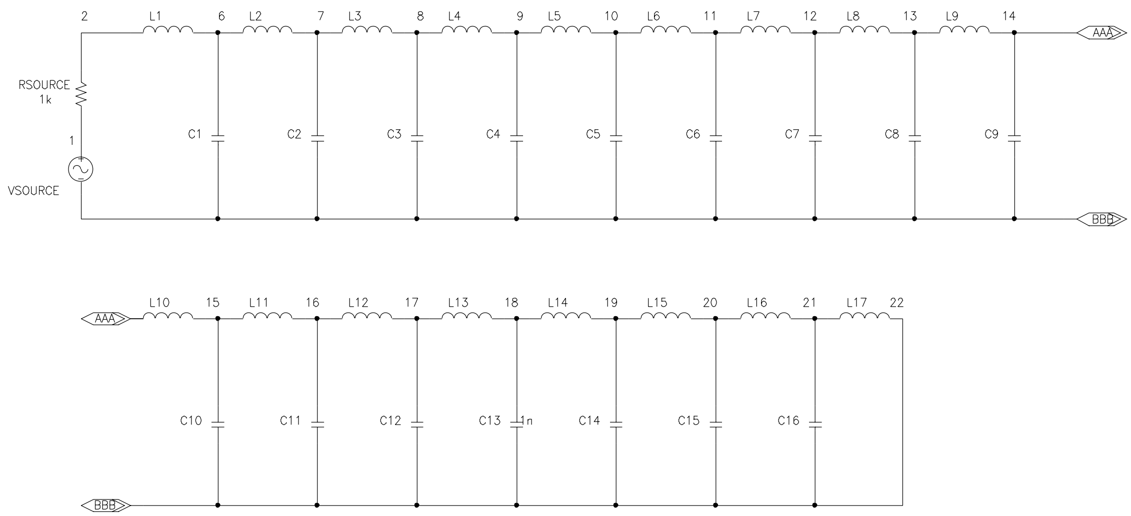
	Claimed invention	Date of patent application	Inventors
1	Improved general-purpose tuning rules for a PID controller	July 12, 2002	Martin A. Keane, John R. Koza, and Matthew J. Streeter
2	Improved general-purpose non-PID	July 12, 2002	Martin A. Keane, John R. Koza, and Matthew J. Streeter

	controllers		

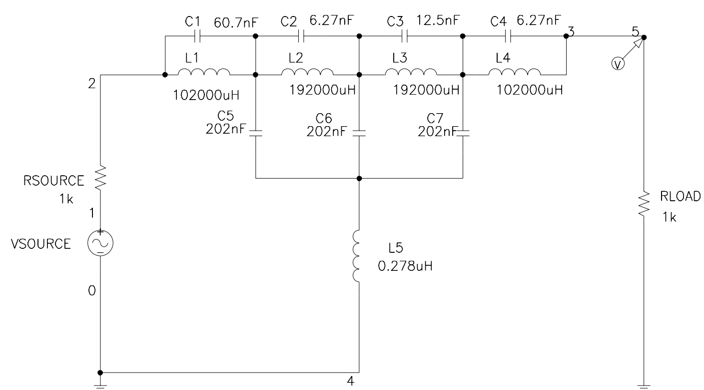


NOVELTY-DRIVEN EVOLUTION

PRIOR ART TEMPLATE



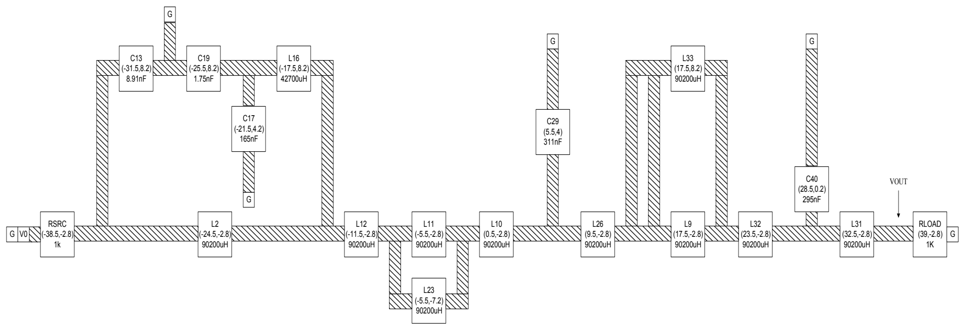
SOLUTION NO. 1



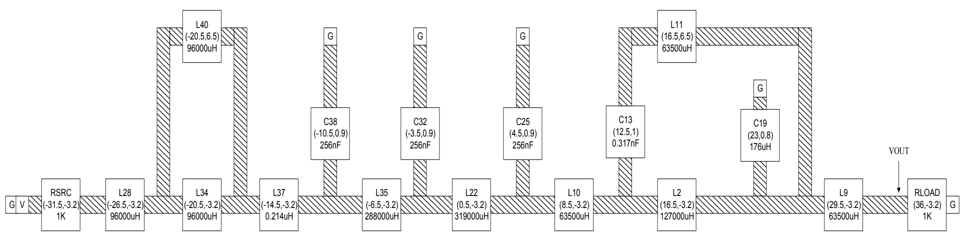
SOLUTION NO. 5

LAYOUT — LOWPASS FILTER 100%-COMPLIANT CIRCUITS

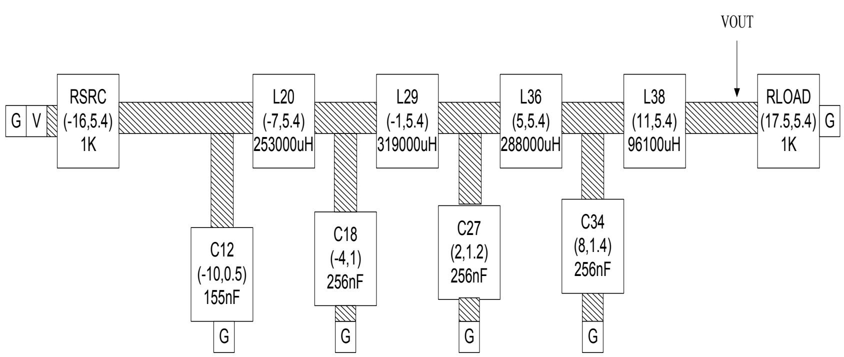
GENERATION 25 WITH 5 CAPACITORS AND 11 INDUCTORS — AREA OF 1775.2



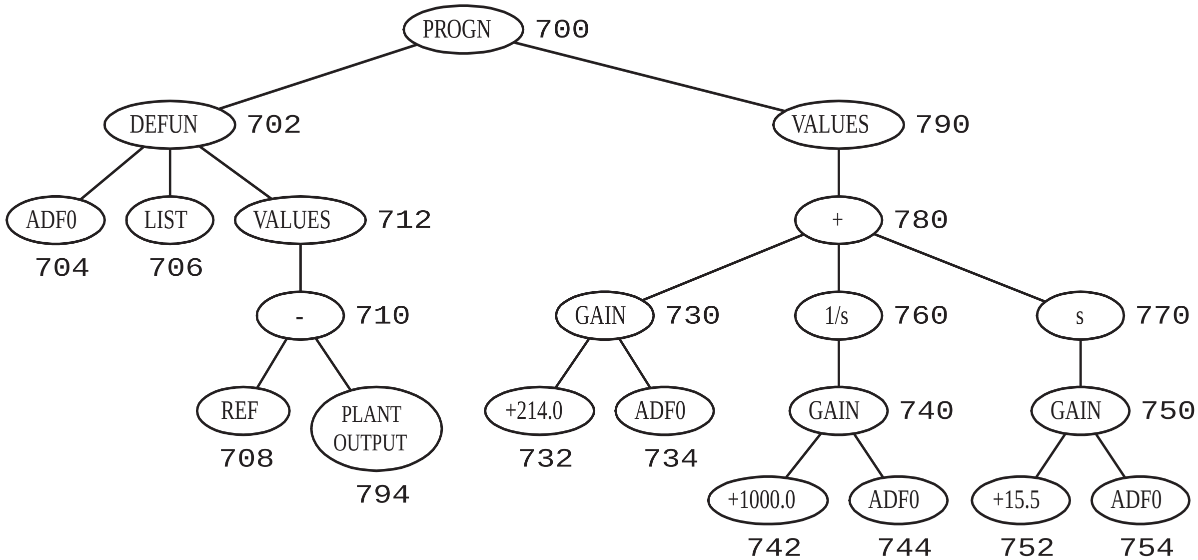
GENERATION 30 WITH 10 INDUCTORS AND 5 CAPACITORS — AREA OF 950.3



BEST-OF-RUN CIRCUIT OF GENERATION 138 WITH 4 INDUCTORS AND 4 CAPACITORS — AREA OF 359.4

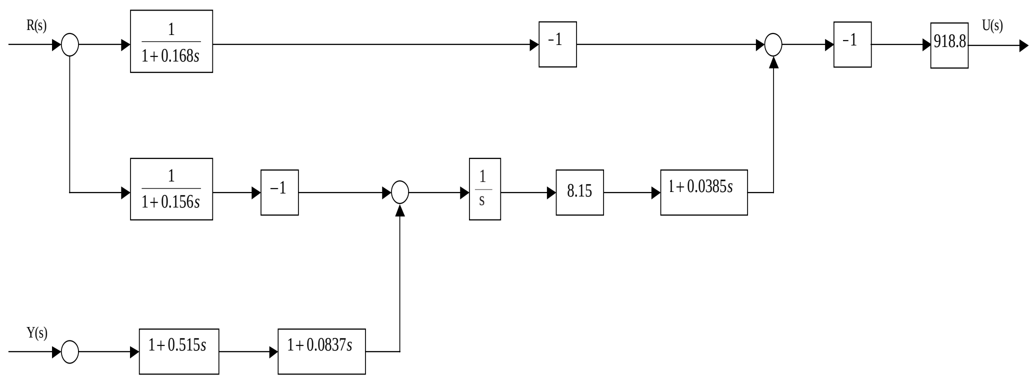


CONTROLLERS



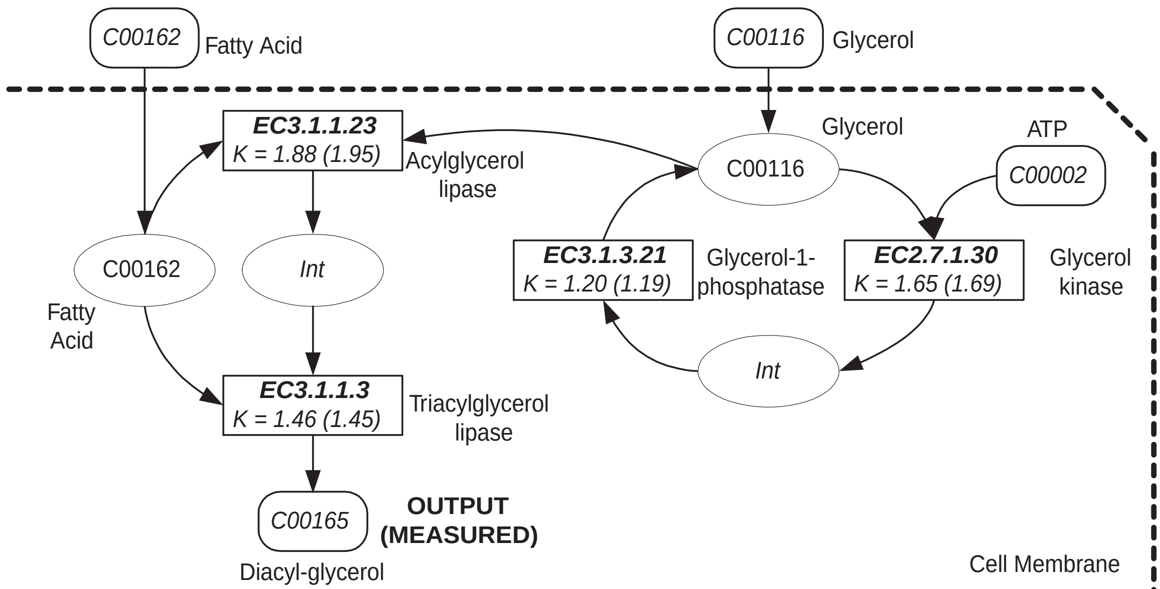
- **ADF can be used for internal feedback**

BEST-OF-RUN GENETICALLY EVOLVED CONTROLLER FOR 2-LAG PLANT

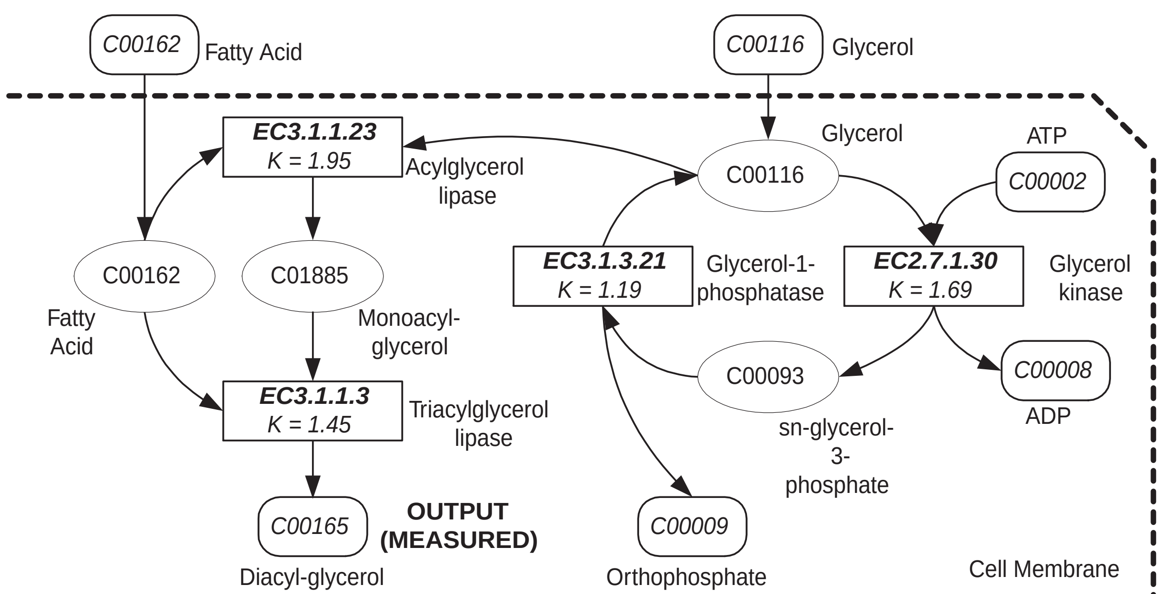


REVERSE ENGINEERING OF METABOLIC PATHWAYS (4-REACTION NETWORK IN PHOSPHOLIPID CYCLE)

BEST-OF-GENERATION 66

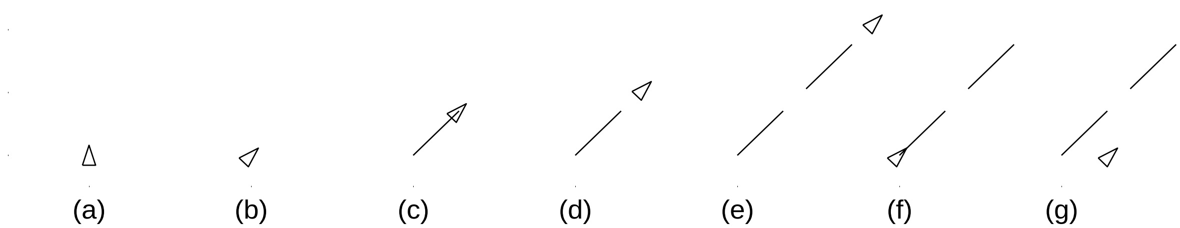


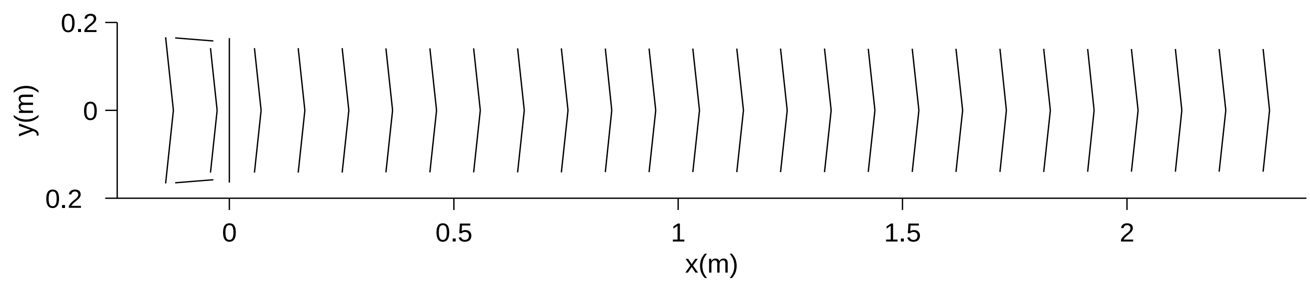
DESIRED



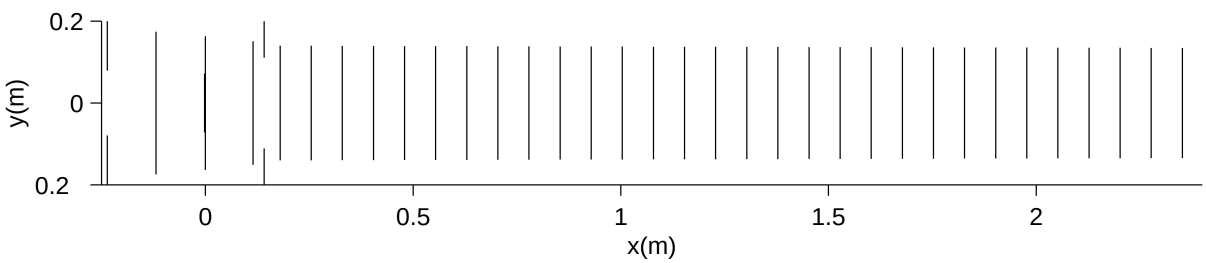
AUTOMATIC SYNTHESIS OF ANTENNA

```
(PROGN3
  (TURN-RIGHT 0.125)
  (LANDMARK
    (REPEAT 2
      (PROGN2
        (DRAW 1.0 HALF-MM-WIRE)
        (DRAW 0.5 NO-WIRE)))
  (TRANSLATE-RIGHT 0.125 0.75))
```





BEST-OF-RUN ANTENNA

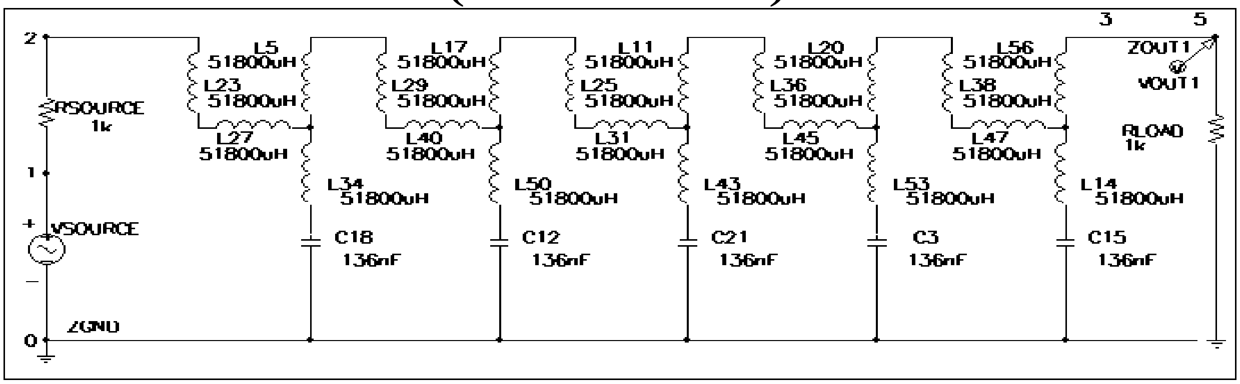


CHARACTERISTICS SUGGESTING THE USE OF GENETIC PROGRAMMING

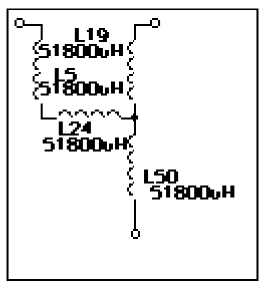
(1) discovering the size and shape of the solution,
(2) reusing substructures,
(3) discovering the number of substructures,
(4) discovering the nature of the hierarchical references among substructures,
(5) passing parameters to a substructure,
(6) discovering the type of substructures (e.g., subroutines, iterations, loops, recursions, or storage),
(7) discovering the number of arguments possessed by a substructure,
(8) maintaining syntactic validity and locality by means of a developmental process, or
(9) discovering a general solution in the form of a parameterized topology containing free variables

REUSE

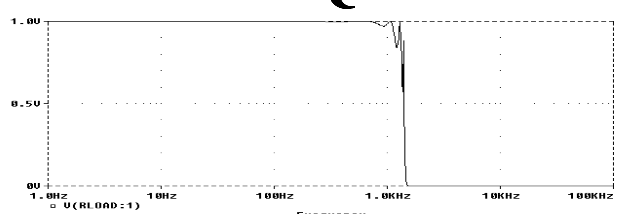
LOWPASS FILTER USING ADFs GENERATION 31 — TOPOLOGY OF CAUER (ELLIPTIC) FILTER



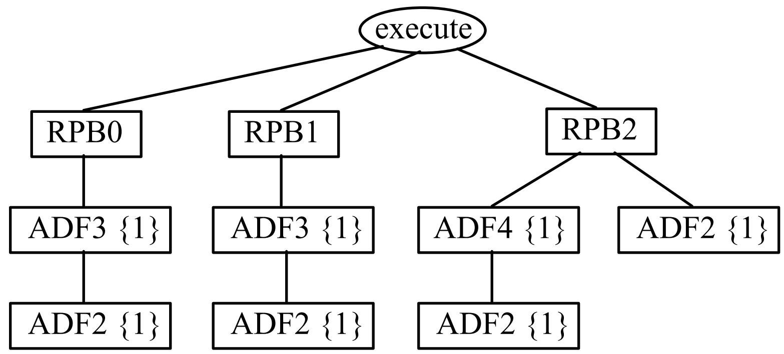
QUINTUPLY-CALLED THREE-PORTED ADF0



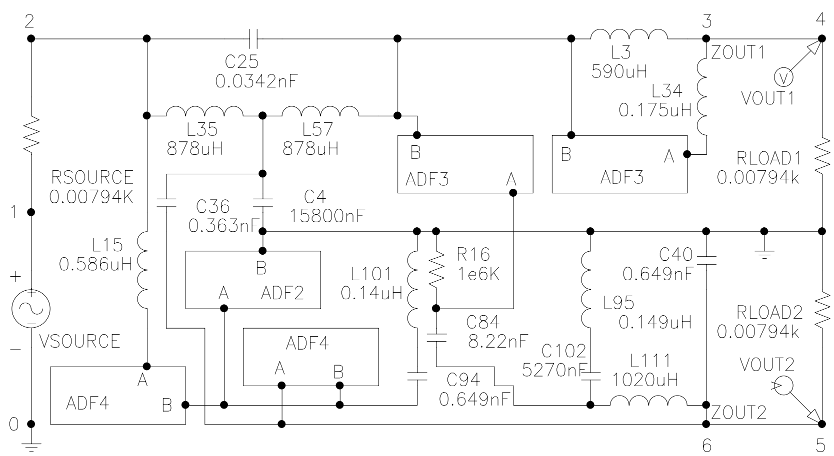
BEHAVIOR IN FREQUENCY DOMAIN



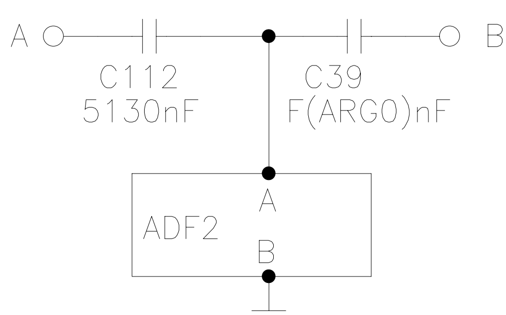
PASSING A PARAMETER TO A SUBSTRUCTURE



BEST-OF-RUN CIRCUIT

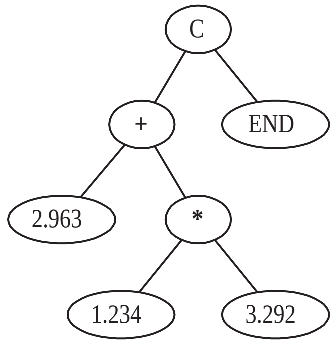


THREE-PORTED ADF3

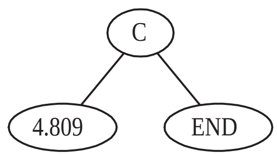


VALUE-SETTING SUBTREES—3 WAYS

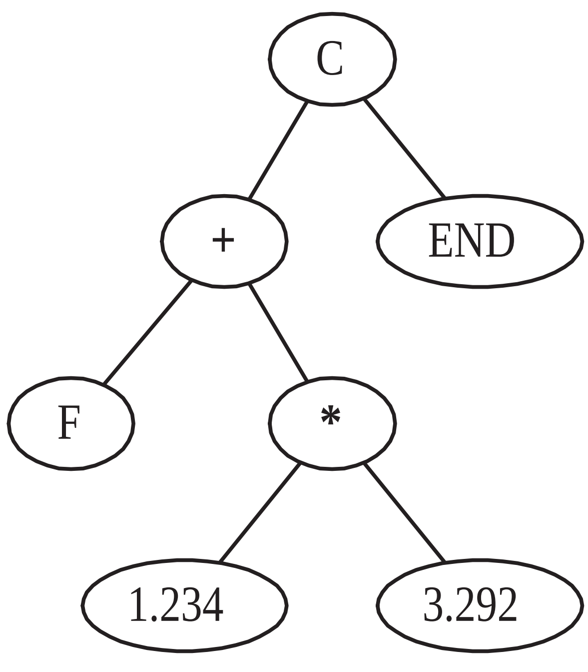
ARITHMETIC-PERFORMING SUBTREE



SINGLE PERTURBABLE CONSTANT

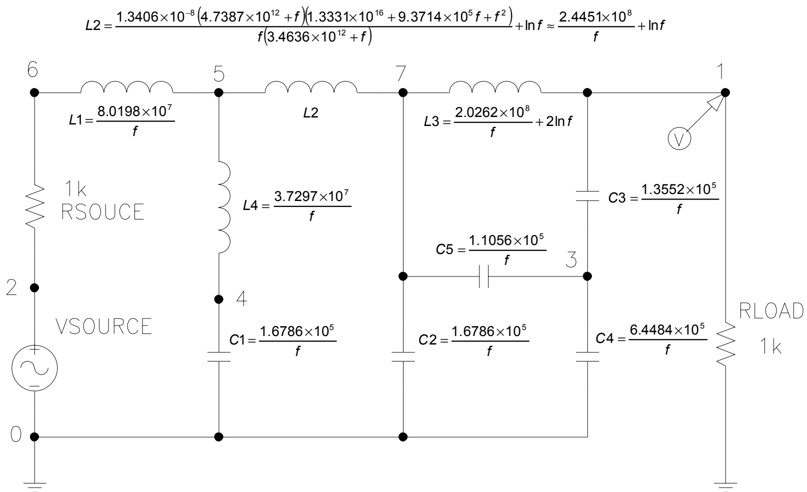


FREE VARIABLE

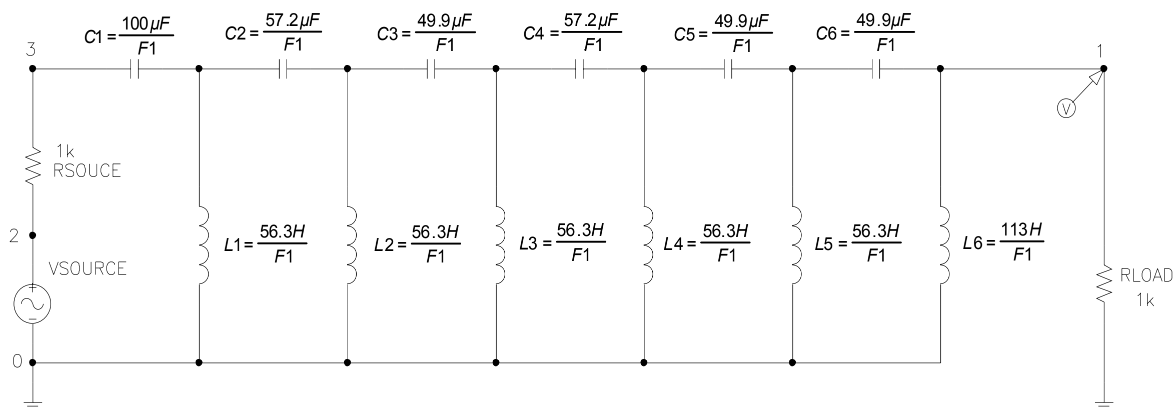


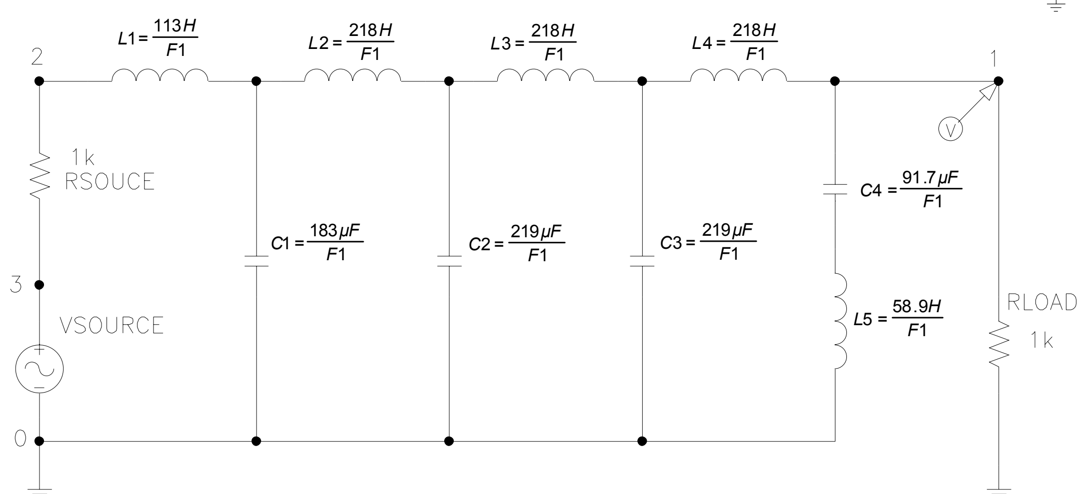
PARAMETERIZED TOPOLOGIES (GIFS)

VARIABLE CUTOFF LOWPASS FILTER

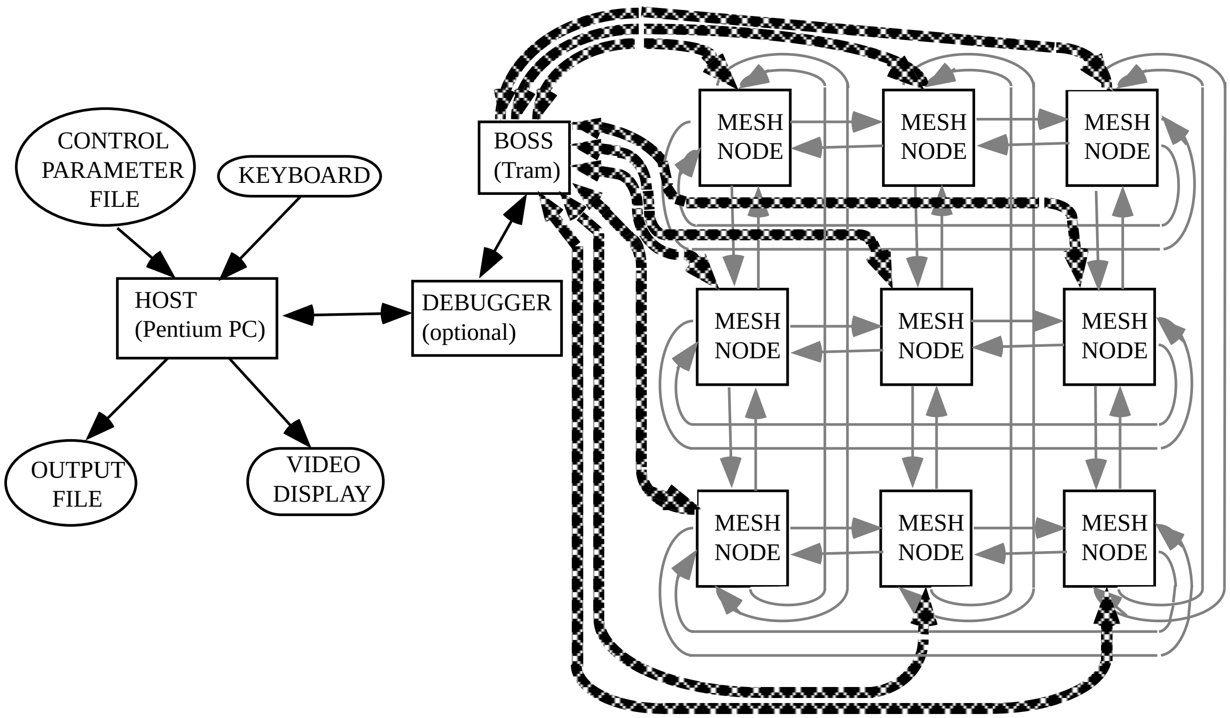


LOWPASS/HIGHPASS FILTER





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)**

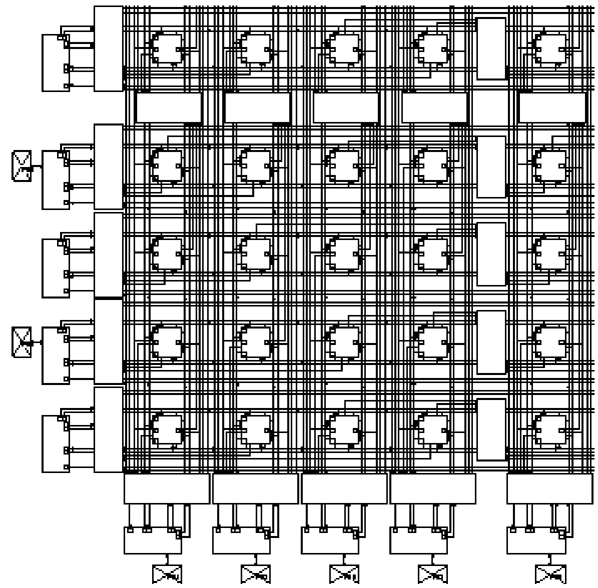
GENETIC PROGRAMMING OVER 15-YEAR PERIOD 1987–2002

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	2000–2002	30.0	9.4	13,900	12

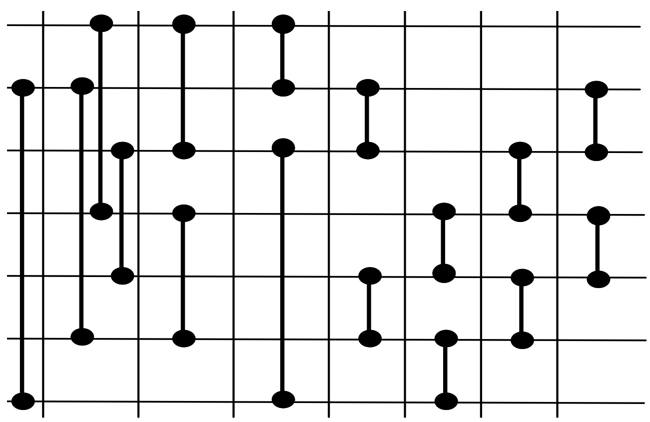
PROGRESSION OF RESULTS

System	Period	Speed-up	Qualitative nature of the results produced by genetic programming
Serial LISP machine	1987–1994	1 (base)	• 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	•Two human-competitive results involving one-dimensional discrete data (not patent-related)
64-node Parsytec parallel machine	1995–2000	22	• 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	• 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	• 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	• Generation of two patentable new inventions

EVOLVABLE HARDWARE USING RAPIDLY RECONFIGURABLE FPGAs



GENETICALLY EVOLVED 7-SORTER



HUMAN-COMPETITIVENESS CRITERIA

	Criterion
A	**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.**
B	**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.**
C	**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.**
D	**The result is publishable in its own right as a new scientific result—independent of the fact that the result was mechanically created.**
E	**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.**
F	**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.**
G	**The result solves a problem of indisputable difficulty in its field.**
H	**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).**

37 HUMAN-COMPETITIVE RESULTS

	Claimed instance	Basis for claim of human-competitiveness	Reference
1	**Creation of a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem**	**B, F**	**Spector, Barnum, and Bernstein 1998**
2	**Creation of a better-than-classical quantum algorithm for Grover's database search problem**	**B, F**	**Spector, Barnum, and Bernstein 1999**
3	**Creation of a quantum algorithm for the depth-two AND/OR query problem that is better than any previously published result**	**D**	**Spector, Barnum, Bernstein, and Swamy 1999; Barnum, Bernstein, and Spector 2000**
4	**Creation of a quantum algorithm for the depth-one OR query problem that is better than any previously published result**	**D**	**Barnum, Bernstein, and Spector 2000**
5	**Creation of a protocol for communicating information through a quantum gate that was previously thought not to permit such communication**	**D**	**Spector and Bernstein 2003**
6	**Creation of a novel variant of quantum dense coding**	**D**	**Spector and Bernstein 2003**
7	**Creation of a soccer-playing program that won its first two games in the Robo Cup 1997 competition**	**H**	**Luke 1998**
8	**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**	**H**	**Andre and Teller 1999**
9	**Creation of four different algorithms for the transmembrane segment identification problem for proteins**	**B, E**	**Sections 18.8 and 18.10 of *GP-2 book* and sections 16.5 and 17.2 of GP-3 book**
10	**Creation of a sorting network for seven items using only 16 steps**	**A, D**	**Sections 21.4.4, 23.6, and 57.8.1 of GP-3 book**

11	**Rediscovery of the Campbell ladder topology for lowpass and highpass filters**	**A, F**	**Section 25.15.1 of GP-3 book and section 5.2 of GP-4 book**
12	**Rediscovery of the Zobel "*M*-derived half section" and "constant *K*" filter sections**	**A, F**	**Section 25.15.2 of GP-3 book**
13	**Rediscovery of the Cauer (elliptic) topology for filters**	**A, F**	**Section 27.3.7 of GP-3 book**
14	**Automatic decomposition of the problem of synthesizing a crossover filter**	**A, F**	**Section 32.3 of GP-3 book**
15	**Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits**	**A, F**	**Section 42.3 of GP-3 book**
16	**Synthesis of 60 and 96 decibel amplifiers**	**A, F**	**Section 45.3 of GP-3 book**
17	**Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions**	**A, D, G**	**Section 47.5.3 of GP-3 book**
18	**Synthesis of a real-time analog circuit for time-optimal control of a robot**	**G**	**Section 48.3 of GP-3 book**
19	**Synthesis of an electronic thermometer**	**A, G**	**Section 49.3 of GP-3 book**
20	**Synthesis of a voltage reference circuit**	**A, G**	**Section 50.3 of GP-3 book**
21	**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**	**D, E**	**Andre, Bennett, and Koza 1996 and section 58.4 of GP-3 book**
22	**Creation of motifs that detect the D–E–A–D box family of proteins and the manganese superoxide dismutase family**	**C**	**Section 59.8 of GP-3 book**
23	**Synthesis of topology for a PID-D2 (proportional, integrative, derivative, and second derivative) controller**	**A, F**	**Section 3.7 of GP-4 book**
24	**Synthesis of an analog circuit equivalent to Philbrick circuit**	**A, F**	**Section 4.3 of GP-4 book**
25	**Synthesis of a `NAND` circuit**	**A, F**	**Section 4.4 of GP-4 book**
26	**Simultaneous synthesis of topology, sizing, placement, and routing of analog electrical circuits**	**A. F, G**	**Chapter 5 of GP-4 book**
27	**Synthesis of topology for a PID (proportional, integrative, and derivative) controller**	**A, F**	**Section 9.2 of GP-4 book**
28	**Rediscovery of negative feedback**	**A, E, F, G**	**Chapter 14 of GP-4 book**
29	**Synthesis of a low-voltage balun circuit**	**A**	**Section 15.4.1 of GP-4 book**
30	**Synthesis of a mixed analog-digital variable capacitor circuit**	**A**	**Section 15.4.2 of GP-4 book**
31	**Synthesis of a high-current load circuit**	**A**	**Section 15.4.3 of GP-4 book**
32	**Synthesis of a voltage-current conversion circuit**	**A**	**Section 15.4.4 of GP-4 book**
33	**Synthesis of a Cubic function generator**	**A**	**Section 15.4.5 of GP-4 book**
34	**Synthesis of a tunable integrated active filter**	**A**	**Section 15.4.6 of GP-4 book**
35	**Creation of PID tuning rules that outperform the Ziegler-Nichols and Åström-Hägglund tuning rules**	**A, B, D, E, F, G**	**Chapter 12 of GP-4 book**
36	**Creation of three non-PID controllers that outperform a PID controller that use Ziegler-Nichols or Åström-Hägglund tuning rules**	**A, B, D, E, F, G**	**Chapter 13 of GP-4 book**
37	**Antenna for NASA Space Technology 5 Mission**	**B, D, E, G**	**Lohn, Hornby, Linden 2004**

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 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**
- **"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**
- **Problem areas where large computerized databases are accumulating and computerized techniques are needed to analyze the data**

FUNDAMENTAL DIFFERENCES BETWEEN GP AND OTHER APPROACHES TO AI AND ML

(1) Representation: Genetic programming overtly conducts it search for a solution to the given problem in program space.

(2) Role of point-to-point transformations in the search: Genetic programming does not conduct its search by transforming a single point in the search space into another single point, but instead transforms a set of points into another set of points.

(3) Role of hill climbing in the search: Genetic programming does not rely exclusively on greedy hill climbing to conduct its search, but instead allocates a certain number of trials, in a principled way, to choices that are known to be inferior.

(4) Role of determinism in the search: Genetic programming conducts its search probabilistically.

(5) Role of an explicit knowledge base: None.

(6) Role of formal logic in the search: None.

(7) Underpinnings of the technique: Biologically inspired.

17 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. 1996b. *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. 1994a. *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. 1999a. *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.

SOME RECENT CONFERENCE PROCEEDINGS

Banzhaf, Wolfgang, Daida, Jason, Eiben, A. E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E. (editors). 1999. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 13-17, 1999, Orlando, Florida USA*. San Francisco, CA: Morgan Kaufmann.

Banzhaf, Wolfgang, Poli, Riccardo, Schoenauer, Marc, and Fogarty, Terence C. 1998. *Genetic Programming: First European Workshop. EuroGP'98. Paris, France*. Lecture Notes in Computer Science. Volume 1391. Berlin, Germany: Springer-Verlag.

Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. Genetic Programming 1996: Proceedings of the First Annual Conference. Cambridge, MA: The MIT Press.

Foster, James A., Lutton, Evelyne, Miller, Julian, Ryan, Conor, and Tettamanzi, Andrea G. B. (editors). 2002. Genetic Programming: 5th European Conference, EuroGP 2002, Kinsale, Ireland, April 2002 Proceedings. Berlin: Springer.

Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L. (editors). 1997. *Genetic Programming 1997: Proceedings of the Second Annual Conference*. San Francisco, CA: Morgan Kaufmann.

Koza, John R., Banzhaf, Wolfgang, Chellapilla, Kumar, Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max H., Goldberg, David E., Iba, Hitoshi, and Riolo, Rick. (editors). 1998. *Genetic Programming 1998: Proceedings of the Third Annual Conference*. San Francisco, CA: Morgan Kaufmann.

Miller, Julian, Tomassini, Marco, Lanzi, Pier Luca, Ryan, Conor, Tettamanzi, Andrea G. B., and Langdon, William B. (editors). 2001. *Genetic Programming: 4th European Conference, EuroGP 2001, Lake Como, Italy, April 2001 Proceedings*. Berlin: Springer.

Poli, Riccardo, Nordin, Peter, Langdon, William B., and Fogarty, Terence C. 1999. *Genetic Programming: Second European Workshop. EuroGP'99. Goteborg, Sweden, May 1999*. Lecture Notes in Computer Science. Volume 1598. Berlin, Germany: Springer-Verlag.

Poli, Riccardo, Banzhaf, Wolfgang, Langdon, William B., Miller, Julian, Nordin, Peter, and Fogarty, Terence C. 2000. *Genetic Programming: European Conference, EuroGP 2000, Edinburgh, Scotland, UK, April 2000, Proceedings*. Lecture Notes in Computer Science. Volume 1802. Berlin, Germany: Springer-Verlag. ISBN 3-540-67339-3.

Riolo, Rich and Worzel, William. 2003. *Genetic Programming: Theory and Practice*. Boston: Kluwer Academic Publishers.

Spector, Lee, Goodman, E., Wu, A., Langdon, William B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, Marco, Pezeshk, S., Garzon, Max, and Burke, E. (editors). 2001. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*. San Francisco, CA: Morgan Kaufmann Publishers. Pages 57 - 65. Whitley, Darrell, Goldberg, David, Cantu-Paz, Erick, Spector, Lee, Parmee, Ian, and Beyer, Hans-Georg (editors). *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference, July 10 - 12, 2000, Las Vegas, Nevada*. San Francisco: Morgan Kaufmann Publishers.

ADVANCES IN GENETIC PROGRAMMING

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

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

GENETIC PROGRAMMING AND EVOLVABLE MACHINES JOURNAL

Editor: Wolfgang Banzhaf

WEB

http://www.genetic-programming.org