THE IMPORTANCE OF REUSE AND DEVELOPMENT IN EVOLVABLE HARDWARE

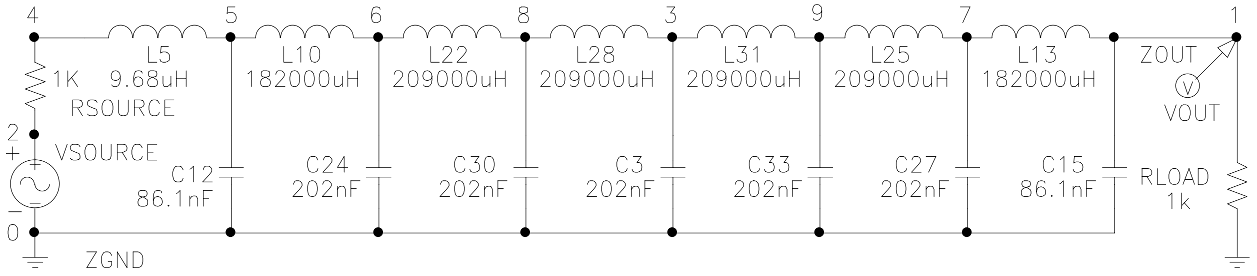
NASA/DoD-EH-2003 — CHICAGO WEDNESDAY JULY 9, 2003

John R. Koza
Stanford University

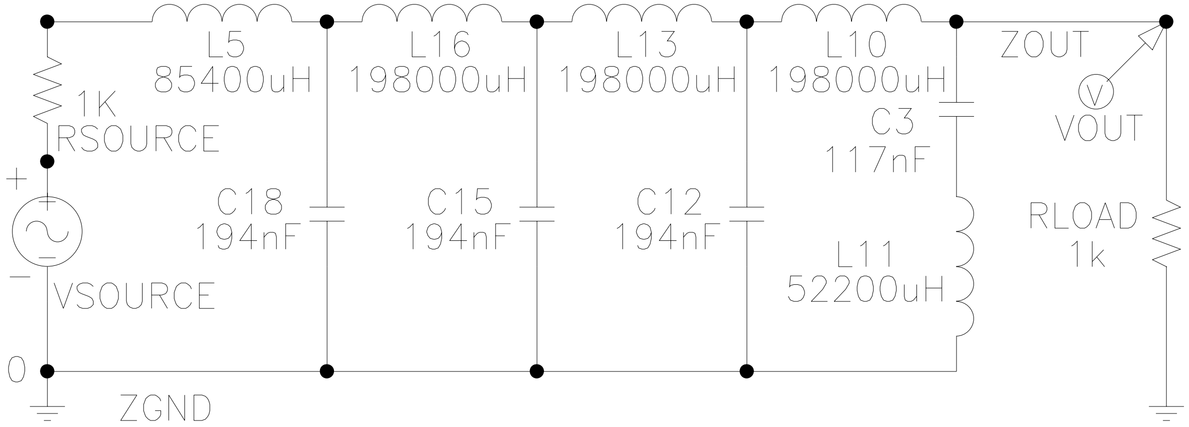
Martin A. Keane
Econometrics Inc.

Matthew J. Streeter
Genetic Programming Inc.

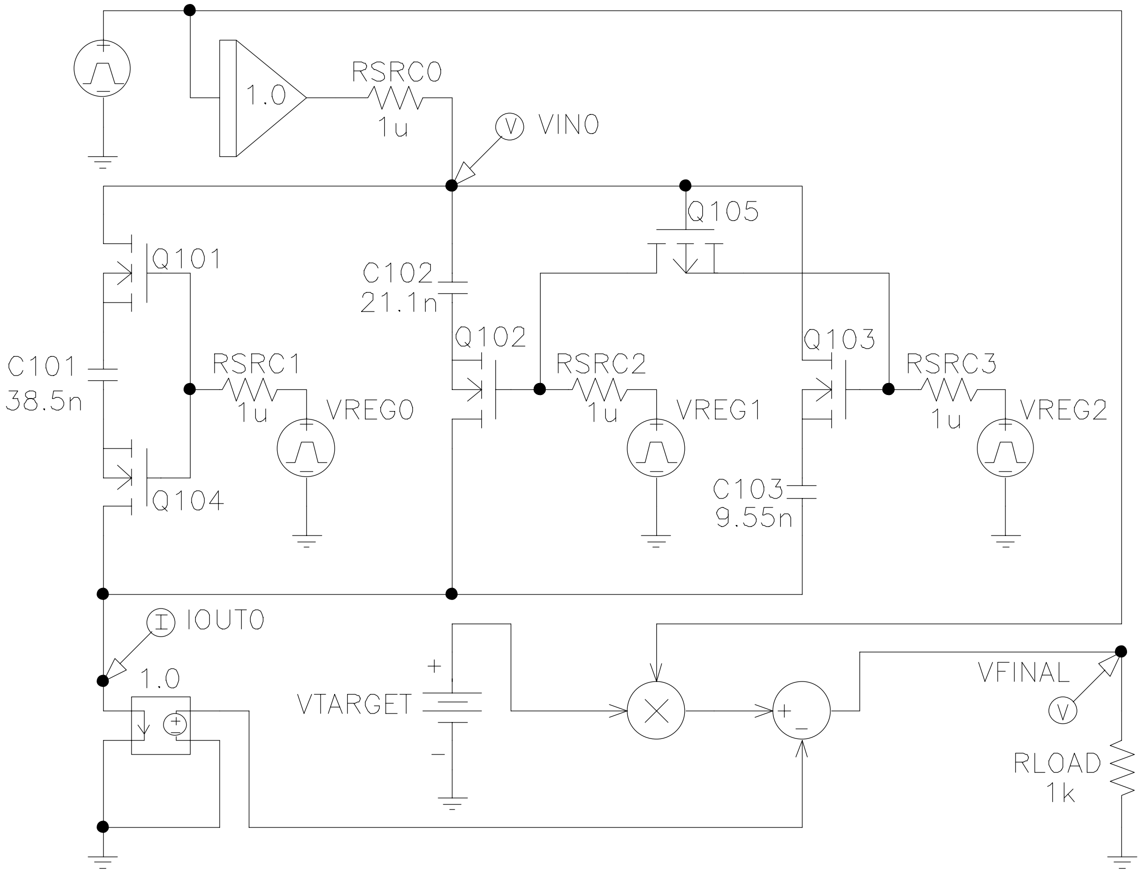
EVOLVED CAMPBELL FILTER
CASCADE OF 6 π SECTIONS
U. S. PATENT 1,227,113



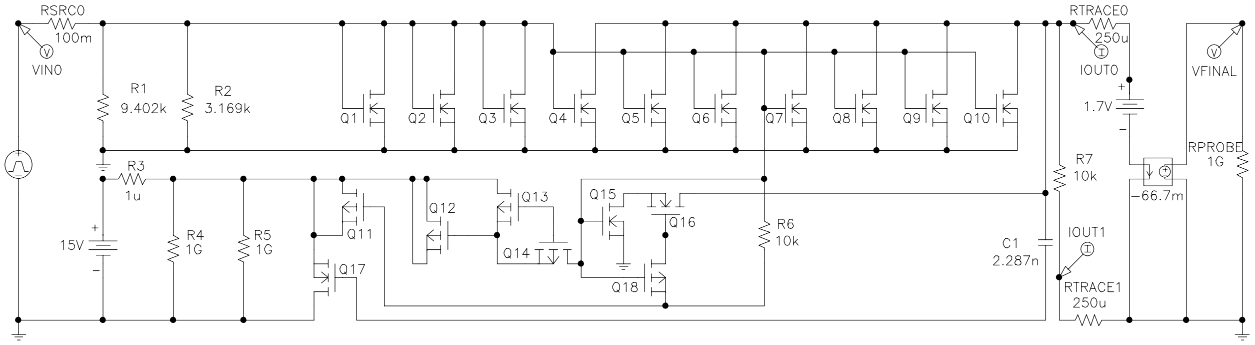
EVOLVED ZOBEL FILTER
CASCADE OF 3 T-SECTIONS
U. S. PATENT 1,538,964



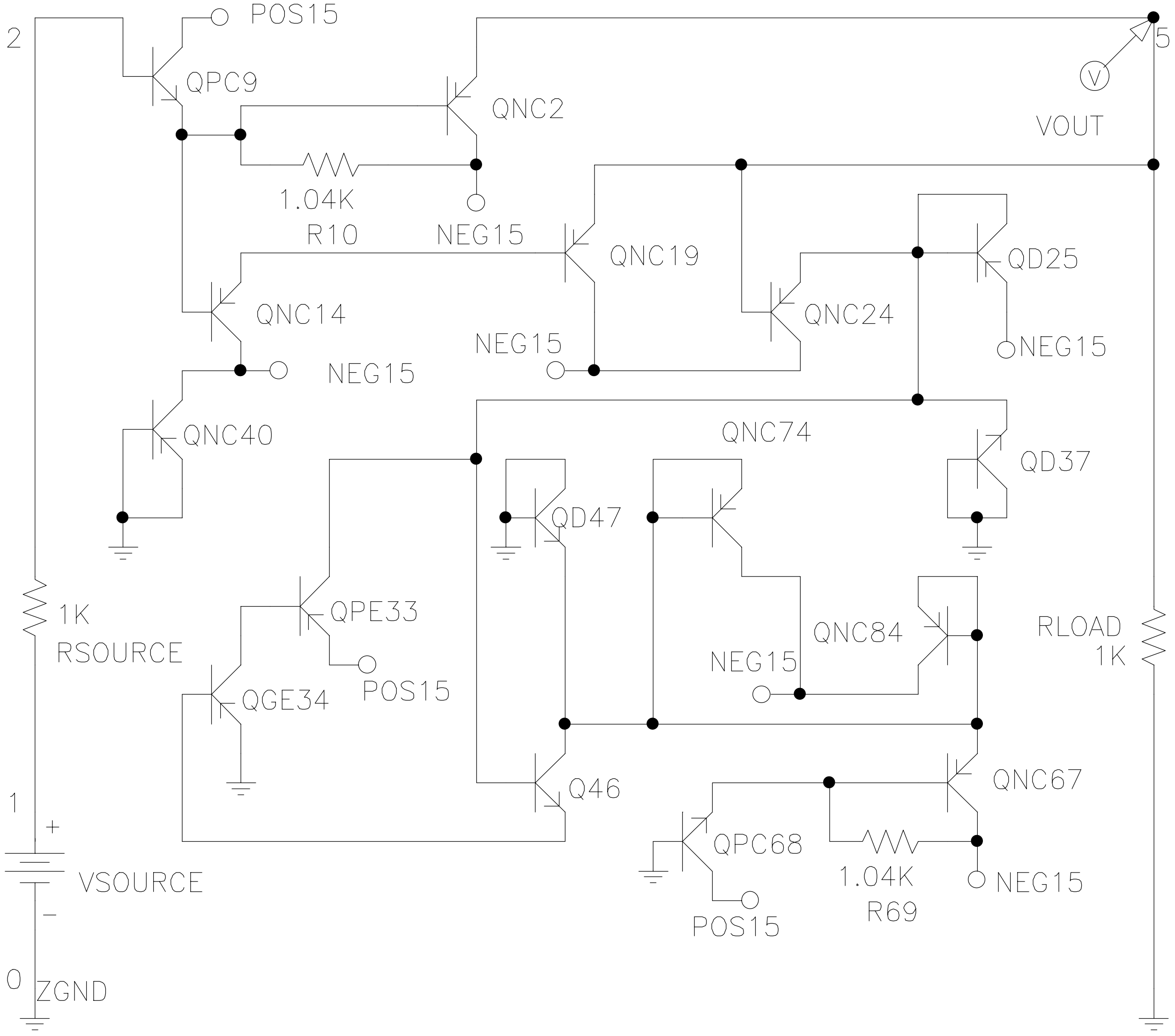
POST-2000 PATENTED INVENTION EVOLVED REGISTER-CONTROLLED CAPACITOR CIRCUIT FROM GENERATION 98



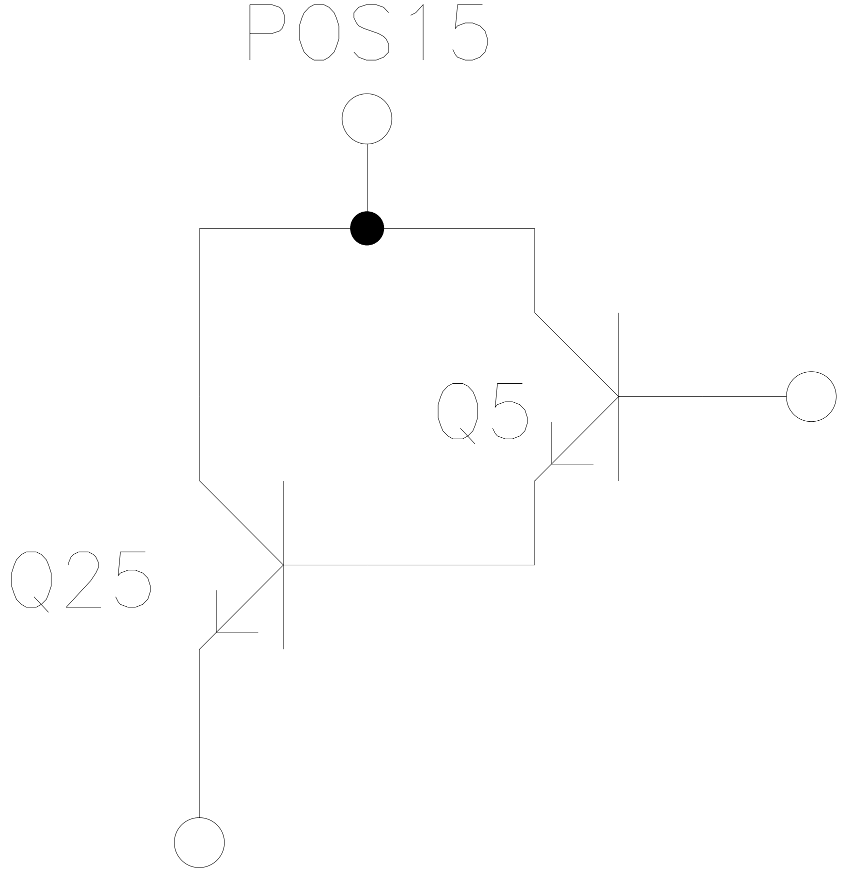
POST-2000 PATENTED INVENTION HIGH CURRENT LOAD CIRCUIT BEST-OF-RUN FROM GENERATION 114



GENETICALLY EVOLVED CUBE ROOT CIRCUIT

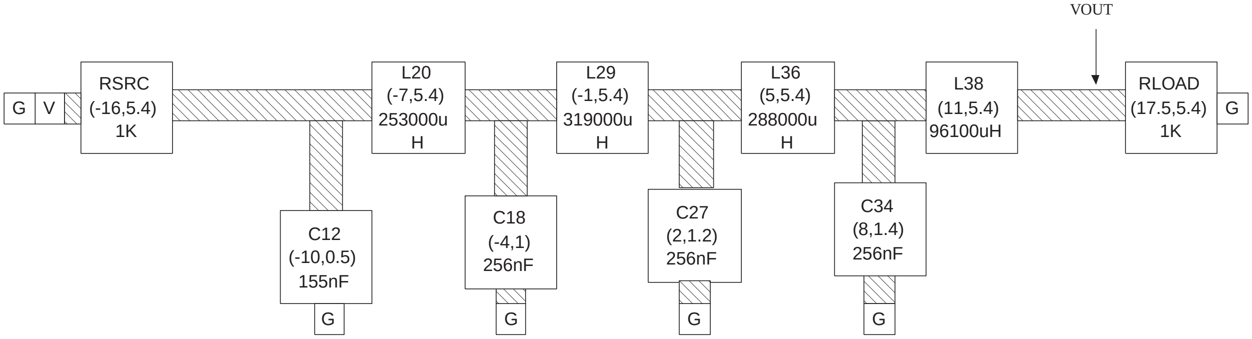


DARLINGTON EMITTER-FOLLOWER FROM GENETICALLY EVOLVED CUBE ROOT CIRCUIT

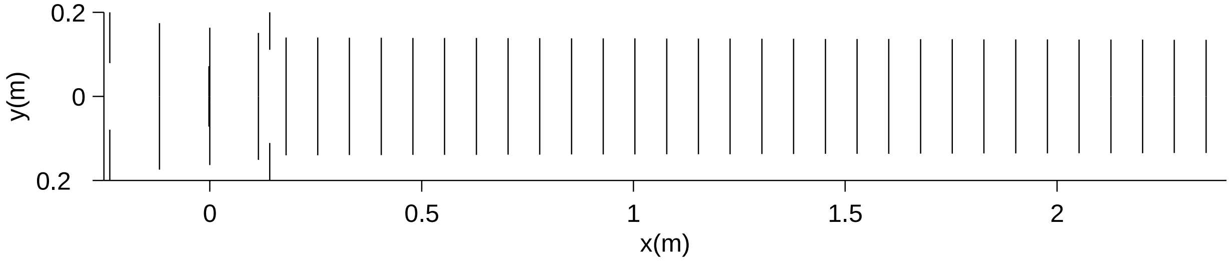


PLACEMENT AND ROUTING (DONE SIMULTANEOUSLY WITH TOPOLOGY AND SIZING)

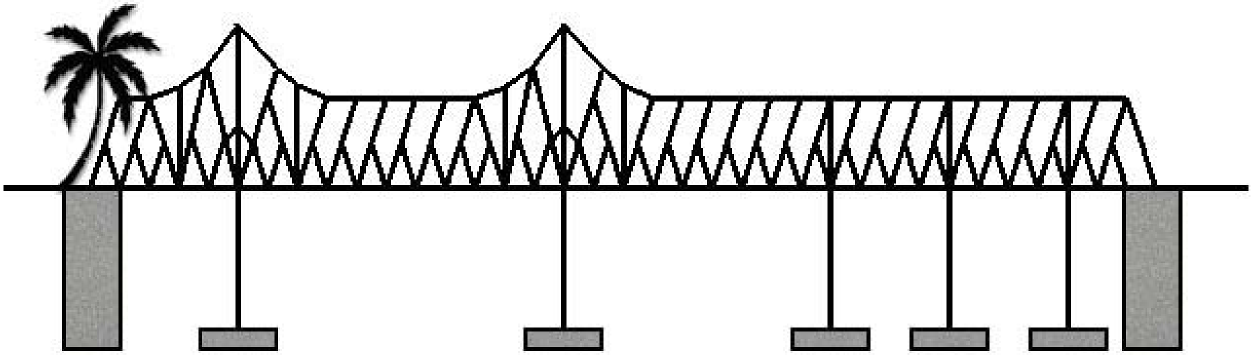
LOWPASS LADDER FILTER



EVOLVED ANTENNA FROM GENERATION 90



OTHER STRUCTURES



MAIN POINTS

- Reuse becomes increasingly important as larger digital and analog circuits are created by evolvable hardware techniques because
  - it avoids the need to reinvent the wheel on each occasion when an already-learned substructure can be used, and
  - thereby accelerating the evolutionary process for problems possessing modularity, symmetry, and regularity and

- A developmental process facilitates reuse
  - facilitates reuse,
  - preserves locality,
  - preserves electrical validity,
  - preserves syntactic validity and executability (thereby alleviating the need for repair and the consequent introduction of genetic material from sources other than the two parents from the current generation of the run)

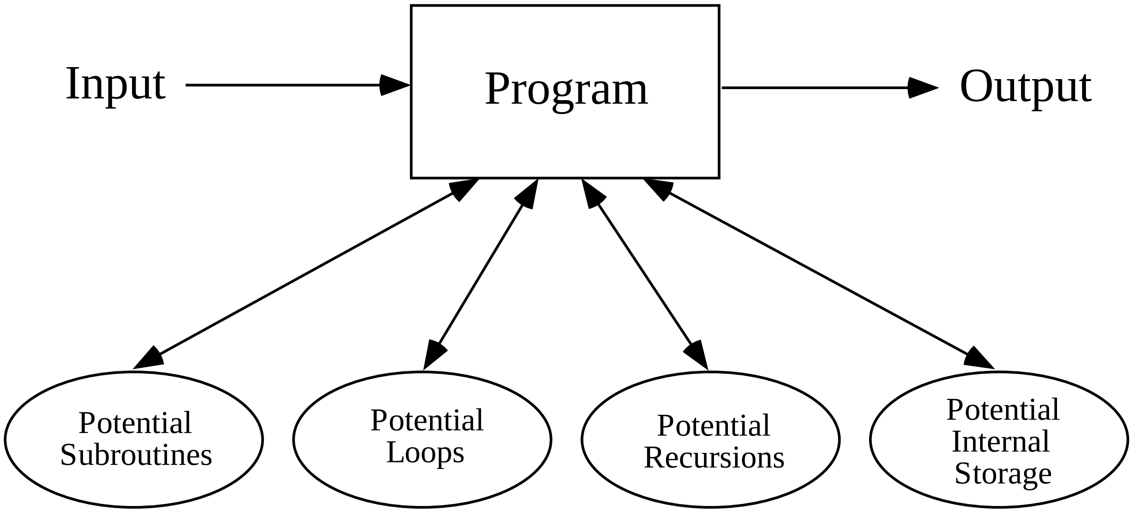
ASPECTS OF REUSE

(1) reusing substructures,
(2) discovering the number of substructures,
(3) discovering hierarchical references among substructures,
(4) passing parameters to a substructure, and
(5) discovering the dimensionality of a substructure (i.e., the number of arguments possessed by the substructure)

3 TYPES OF REUSE

- **The circuit-constructing functions responsible for a useful subcircuit may reside in an automatically defined function (ADF)**
- **The circuit-constructing functions responsible for a useful subcircuit may reside in an automatically defined copy (ADC)**
- **Although often overlooked, the ordinary crossover operation often acts as a mechanism for reusing a subtree that is responsible for a useful subcircuit**

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)

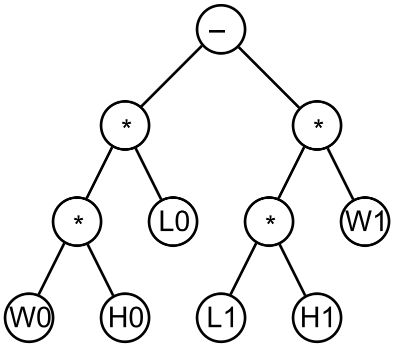
PROBLEM OF SYMBOLIC REGRESSION INVOLVING DEPENDENT VARIABLE, D, AND 6 INDEPENDENT VARIABLES

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**

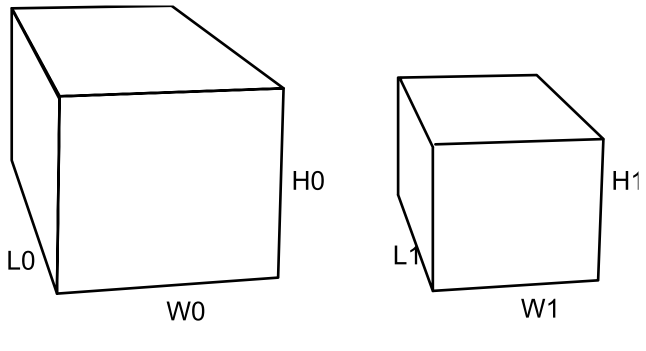
AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

SOLUTION WITHOUT ADFs

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



$$D = W0*L0*H0 - W1*L1*H1$$



AUTOMATICALLY DEFINED FUNCTIONS (ADFS, SUBROUTINES)

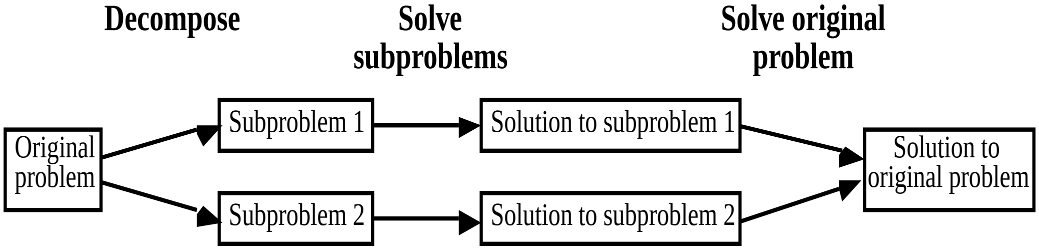
IF WE ADD TWO NEW VARIABLES FOR VOLUME (V_0 AND V_1), THE 6-DIMENSIONAL NON-LINEAR REGRESSION PROBLEM BECOMES AN 8-DIMENSIONAL PROBLEM

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**

AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

TOP-DOWN VIEW OF THREE STEP HIERARCHICAL PROBLEM-SOLVING PROCESS

DIVIDE AND CONQUER



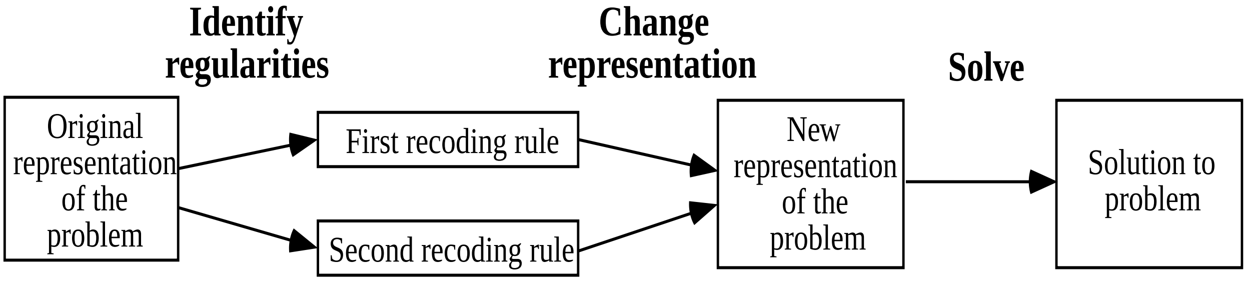
- **Decompose a problem into subproblems**

- **Solve the subproblems**

- **Assemble the solutions of the subproblems into a solution for the overall problem**

AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

BOTTOM-UP VIEW OF THREE STEP HIERARCHICAL PROBLEM-SOLVING PROCESS

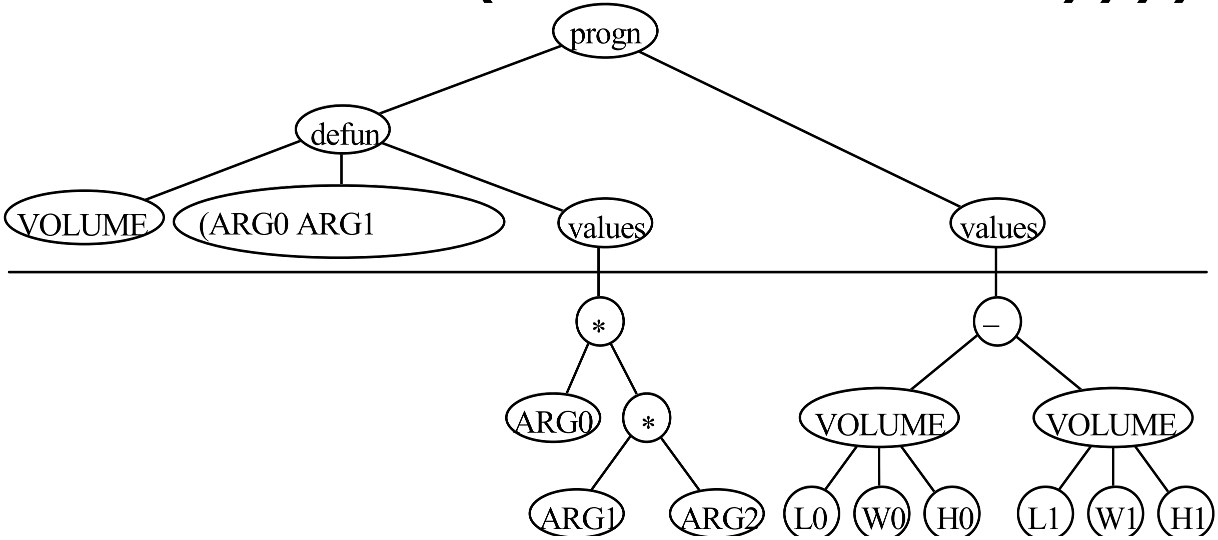


- **Identify regularities**
- **Change the representation**
- **Solve the overall problem**

AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

AN OVERALL COMPUTER PROGRAM CONSISTING OF ONE FUNCTION-DEFINING BRANCH (ADF, SUBROUTINE) AND ONE RESULT-PRODUCING BRANCH (MAIN PROGRAM)

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



8 MAIN POINTS FROM *GENETIC PROGRAMMING II: AUTOMATIC DISCOVERY OF REUSABLE PROGRAMS*

- **ADFs work.**
- **ADFs do not solve problems in the style of human programmers.**
- **<u>ADFs reduce the computational effort required to solve a problem.</u>**
- **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**
  - **bit positions in a digital register**

REUSE

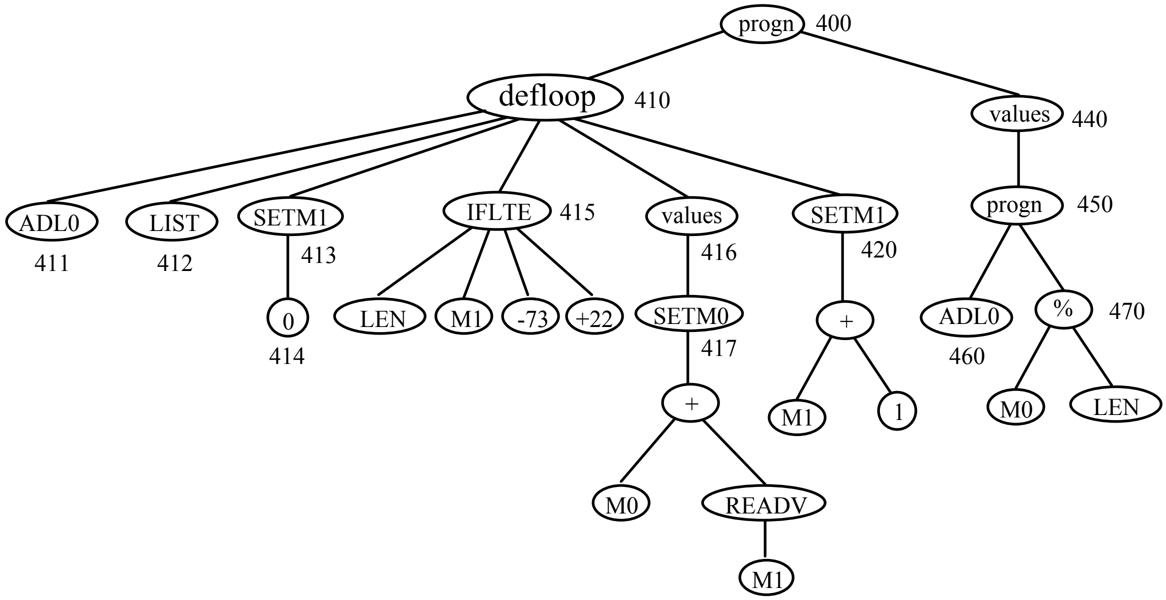
TRANSMEMBRANE SEGMENT IDENTIFICATION PROBLEM

GENERATION 20 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

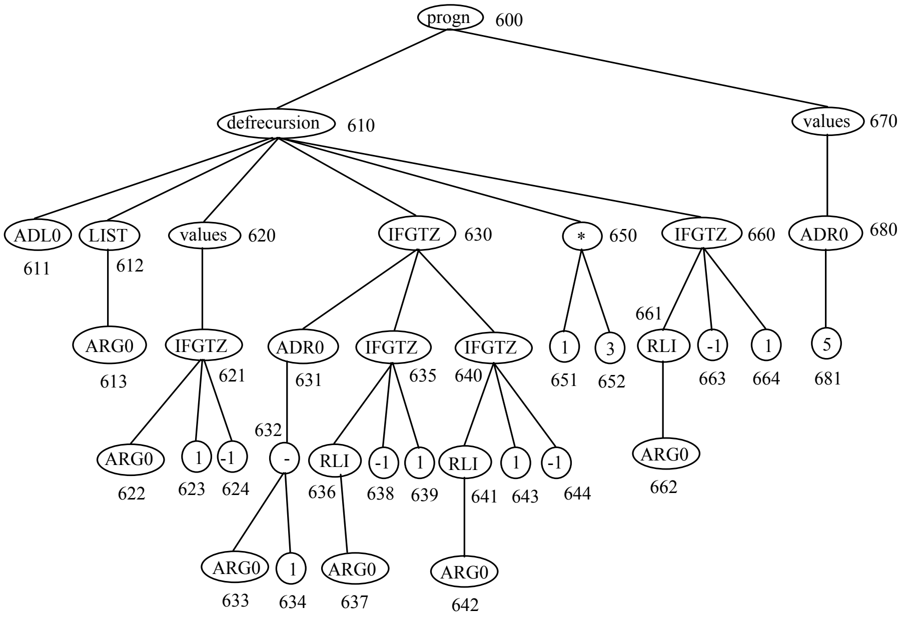
EXAMPLE OF A PROGRAM WITH A FOUR-BRANCH AUTOMATICALLY DEFINED LOOP (ADL0) AND A RESULT-PRODUCING BRANCH



REUSE

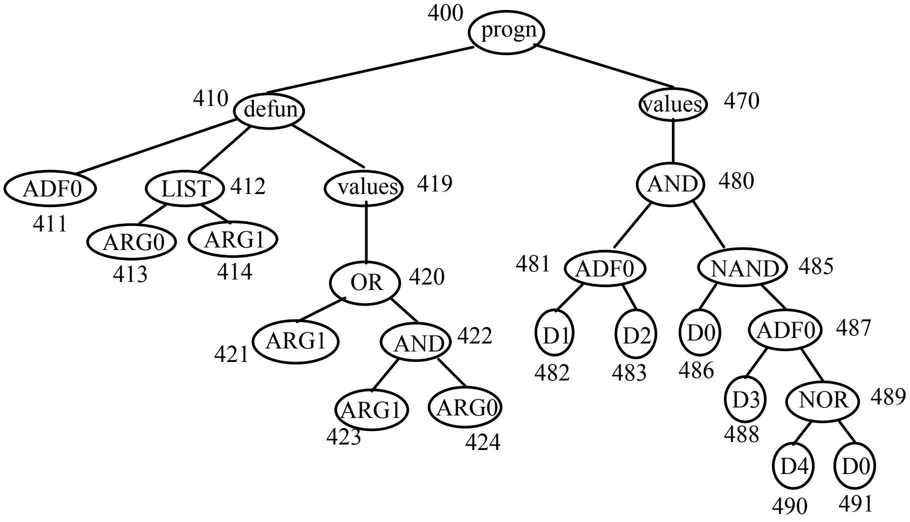
AUTOMATICALLY DEFINED RECURSION (ADR0) AND A RESULT-PRODUCING BRANCH

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



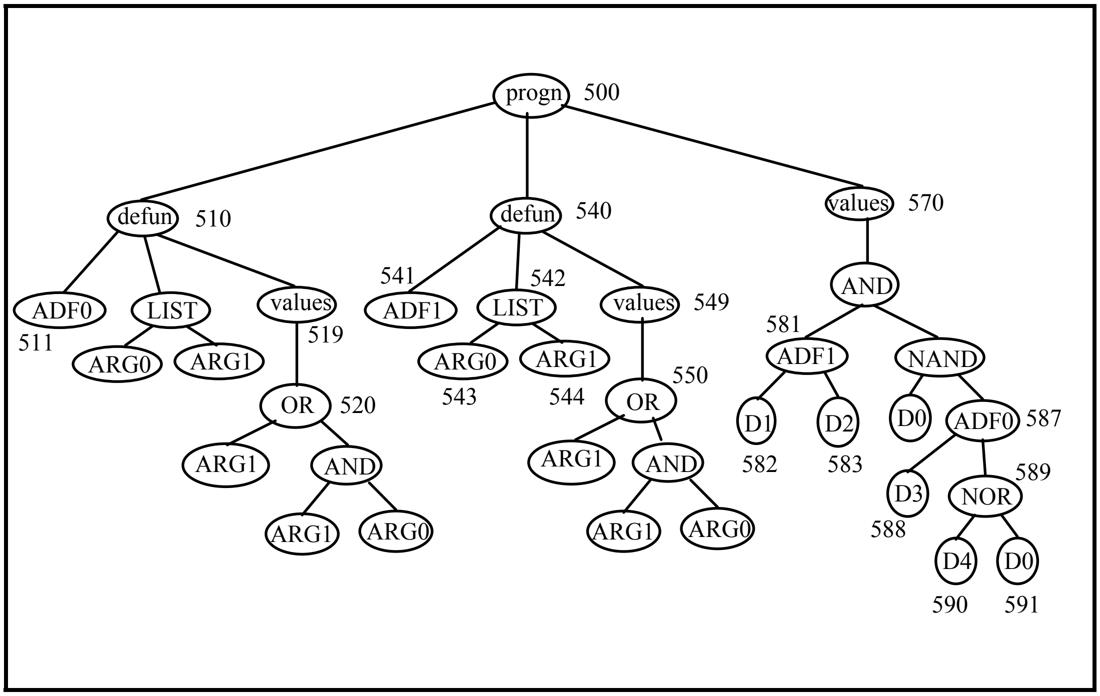
ARCHITECTURE-ALTERING OPERATIONS

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



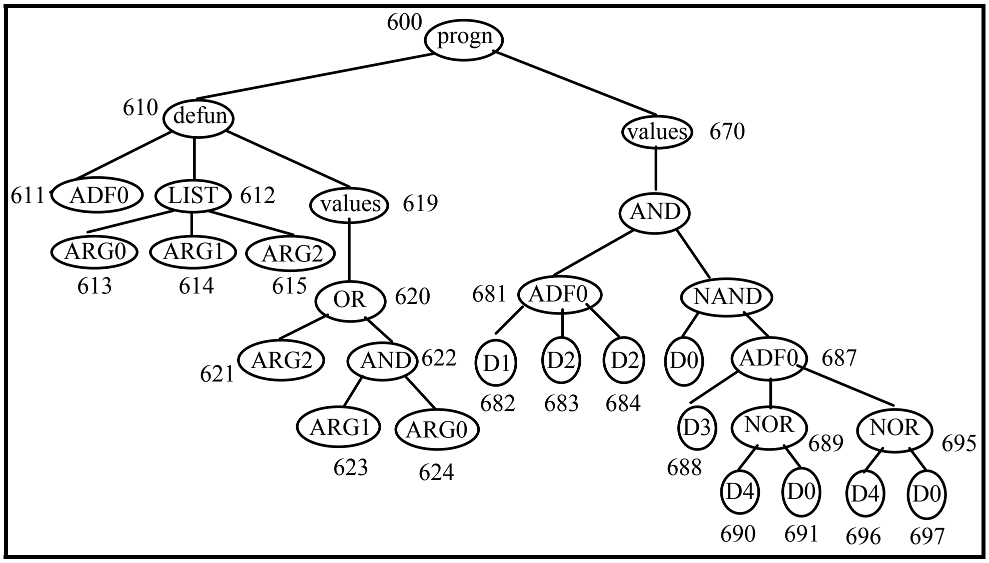
ARCHITECTURE-ALTERING OPERATIONS

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



ARCHITECTURE-ALTERING OPERATIONS

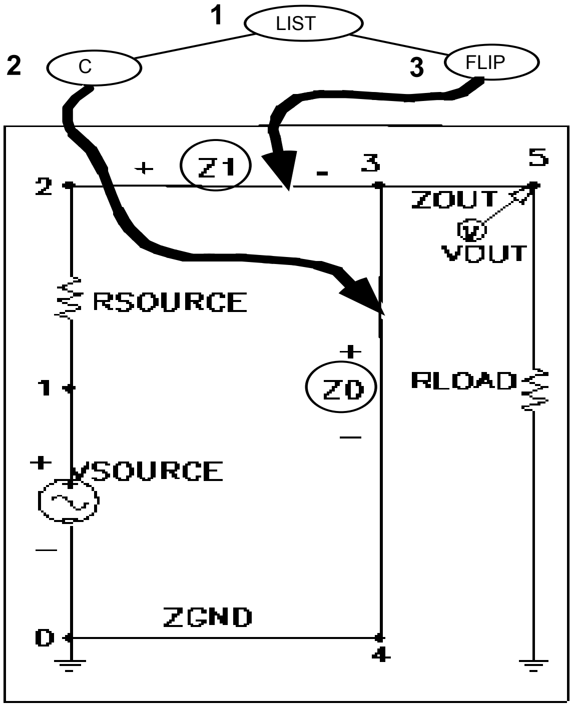
PROGRAM WITH ARGUMENT MAP OF {3} CREATED USING THE OPERATION OF ARGUMENT DUPLICATION



DEVELOPMENTAL GP

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.

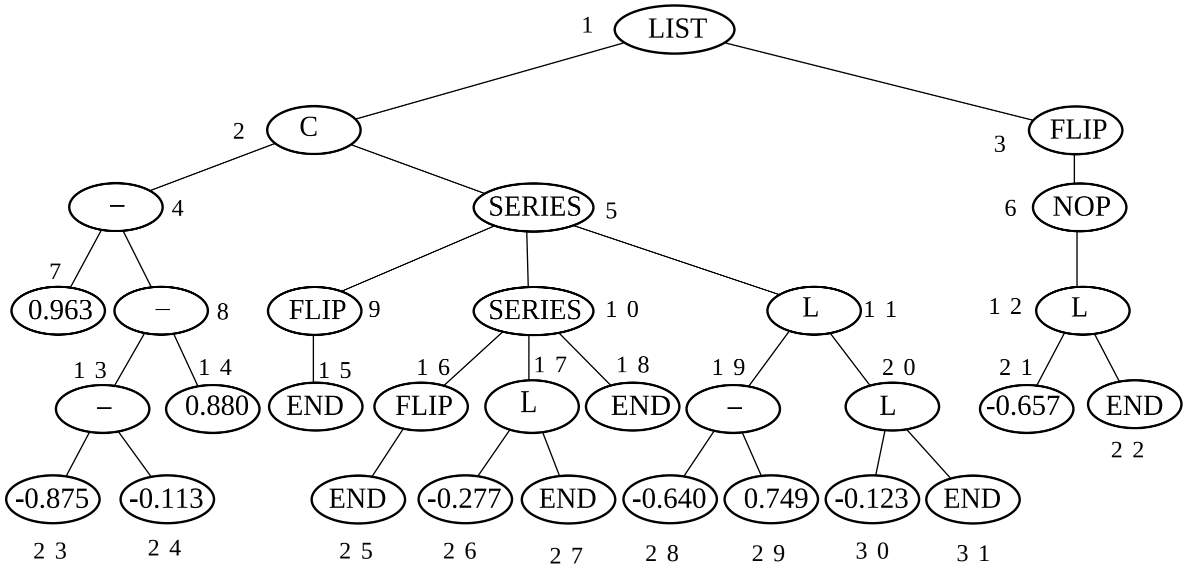
DEVELOPMENTAL GP

THE INITIAL CIRCUIT

- **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

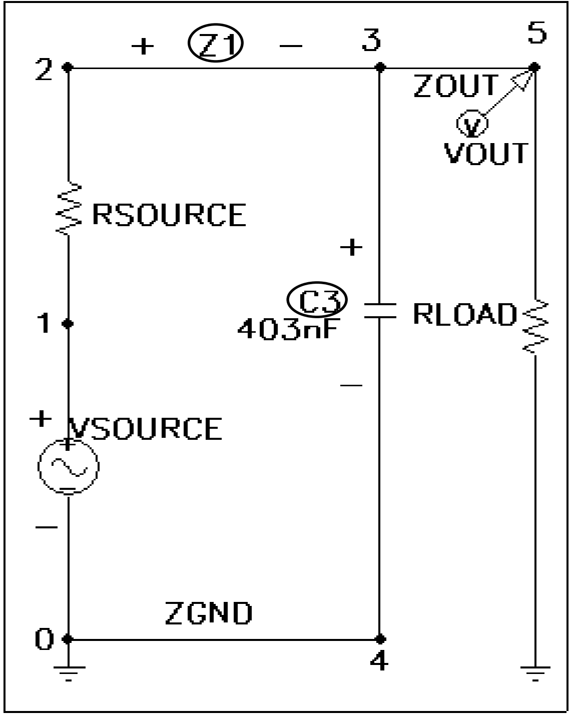
CIRCUIT FROM A CIRCUIT-CONSTRUCTING PROGRAM TREE AND LISP S-EXPRESSION



```
(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

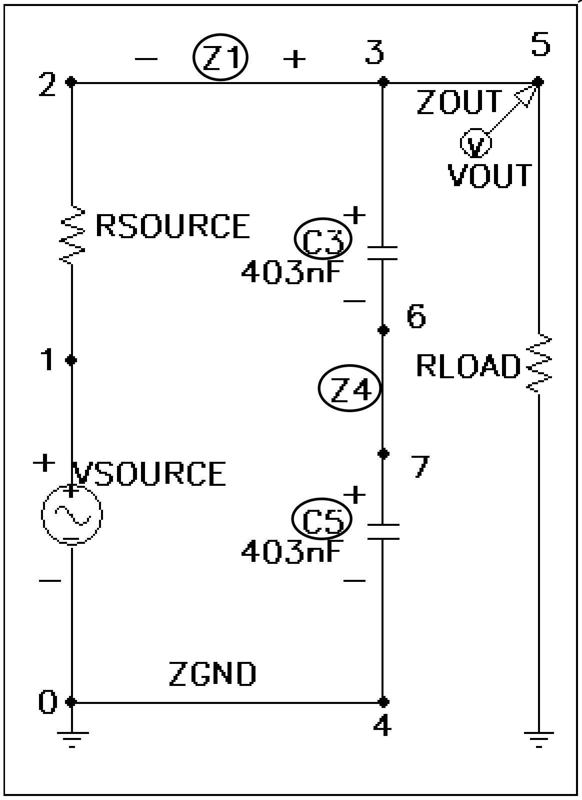
RESULT OF COMPONENT-CREATING FUNCTION (CAPACITOR-INSERTING C FUNCTION) AND ITS VALUE-SETTING SUBTREE (ARITHMETIC-PERFORMING SUBTREE)



```
(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)))))
```

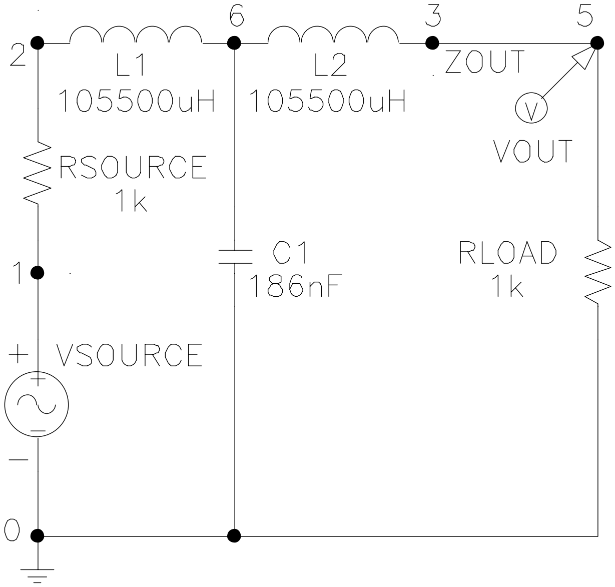
DEVELOPMENTAL GP

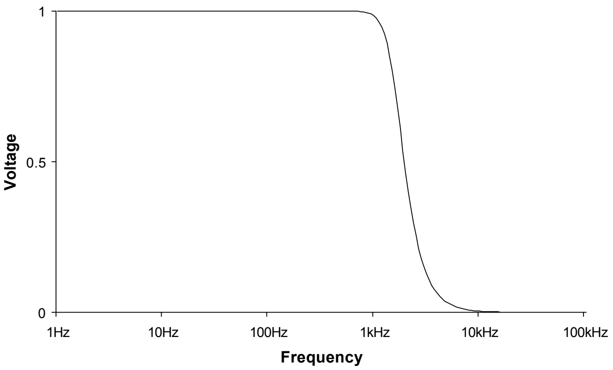
RESULT OF TOPOLOGY MODIFYING FUNCTION (SERIES DIVISION FUNCTION SERIES)



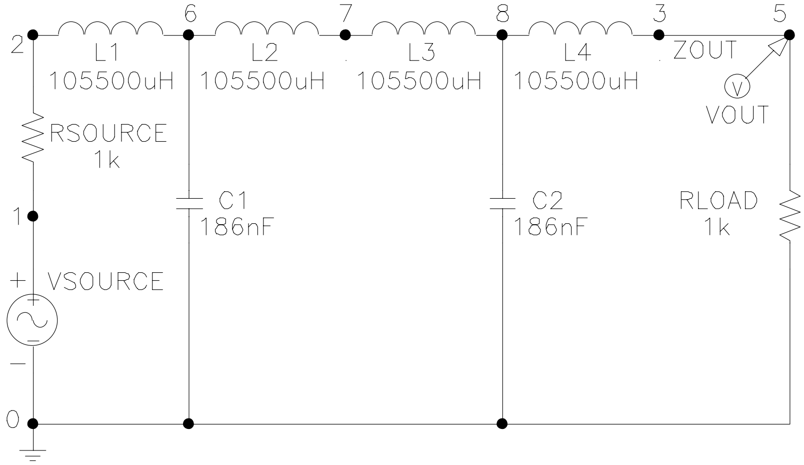
(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (**series** (flip end) (series (flip end) (L -0.277 end) end) (L (– -0.640 0.749) (L -0.123 end)))) (flip (nop (L -0.657 end)))))

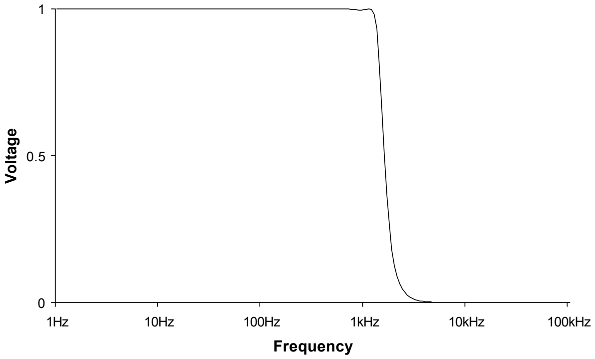
CIRCUIT CONSISTING OF ONE T-SECTION



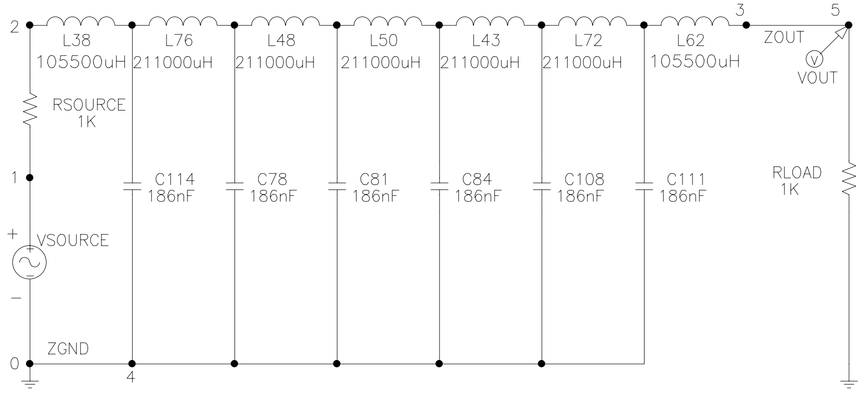


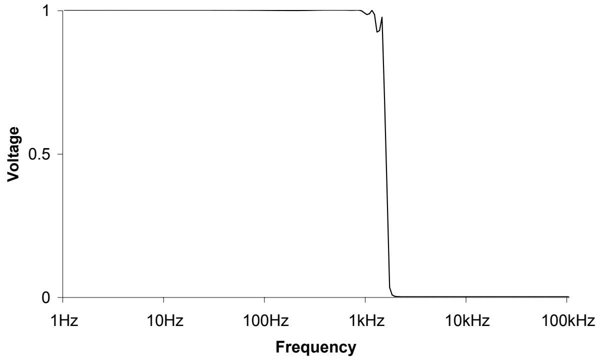
CIRCUIT CONSISTING OF TWO T-SECTIONS





CIRCUIT CONSISTING OF SIX T-SECTIONS





A FIRST ILLUSTRATIVE CHROMOSOME FOR ONE T-SECTION

L	**106**	**2**	**6**	C	**186**	**6**	**0**	L	**106**	**6**	**3**	W	**0**	**0**	**7**	C	**100**	**0**	**7**	C	**200**	**8**	**0**

CHROMOSOME FOR CIRCUIT CONSISTING OF TWO T-SECTIONS

L	**106**	**2**	**6**	C	**186**	**6**	**0**	L	**106**	**6**	**7**	L	**106**	**7**	**8**	C	**186**	**8**	**0**	L	**106**	**8**	**3**

SUB-STRING PRODUCING A T-SECTION

L	**106**	**2**	**6**	C	**186**	**6**	**0**	L	**106**	**6**	**3**

CHROMOSOME OF FIRST PARENT

L	**106**	**2**	**6**	W	**0**	**6**	**7**	C	**186**	**7**	**0**	W	**0**	**7**	**8**	L	**106**	**8**	**9**	W	**0**	**9**	**3**

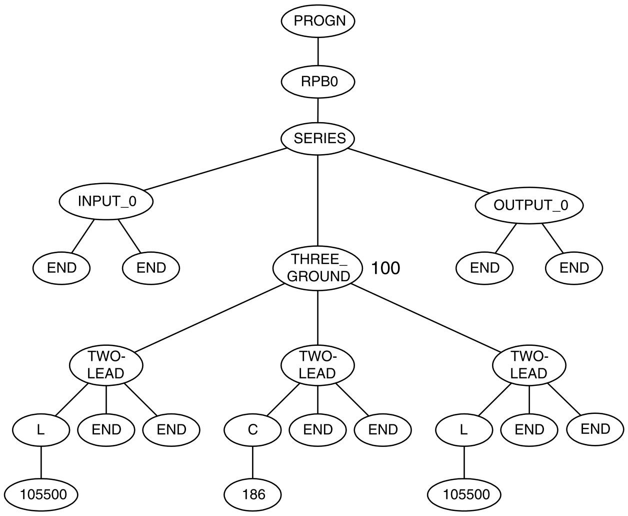
CHROMOSOME OF SECOND PARENT

W	**0**	**8**	**6**	W	**0**	**9**	**3**	W	**0**	**7**	**8**	L	**106**	**6**	**9**	C	**186**	**8**	**0**	L	**106**	**2**	**7**

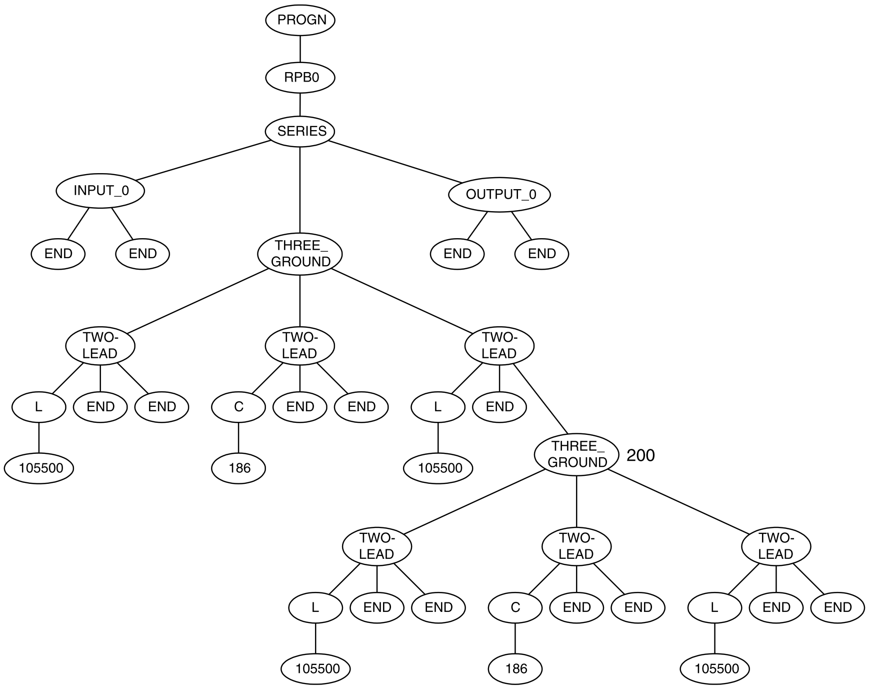
RESULT OF A CROSSOVER

L	**106**	**2**	**6**	W	**0**	**6**	**7**	C	**186**	**7**	**0**	L	**106**	**6**	**9**	C	**186**	**8**	**0**	L	**106**	**2**	**7**

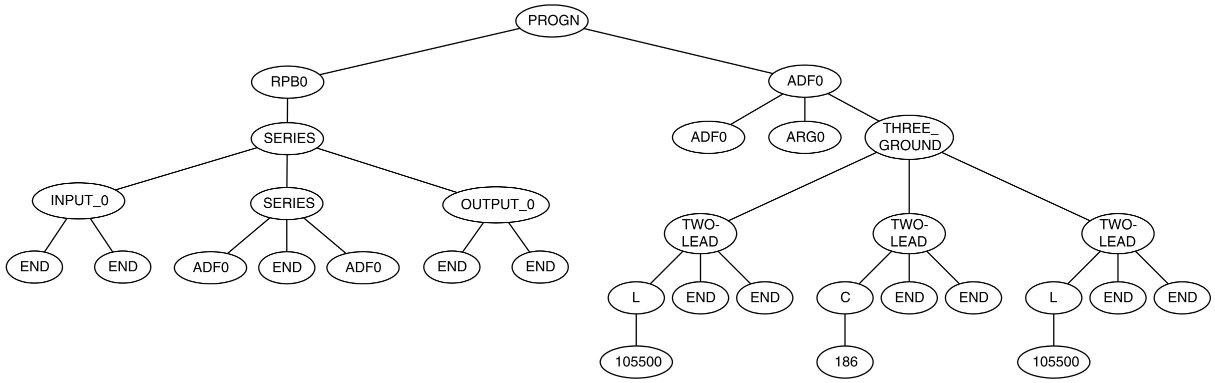
A CIRCUIT-CONSTRUCTING PROGRAM TREE THAT DEVELOPS INTO THE SINGLE T-SECTION



A CIRCUIT-CONSTRUCTING PROGRAM TREE THAT DEVELOPS INTO THE TWO T-SECTIONS



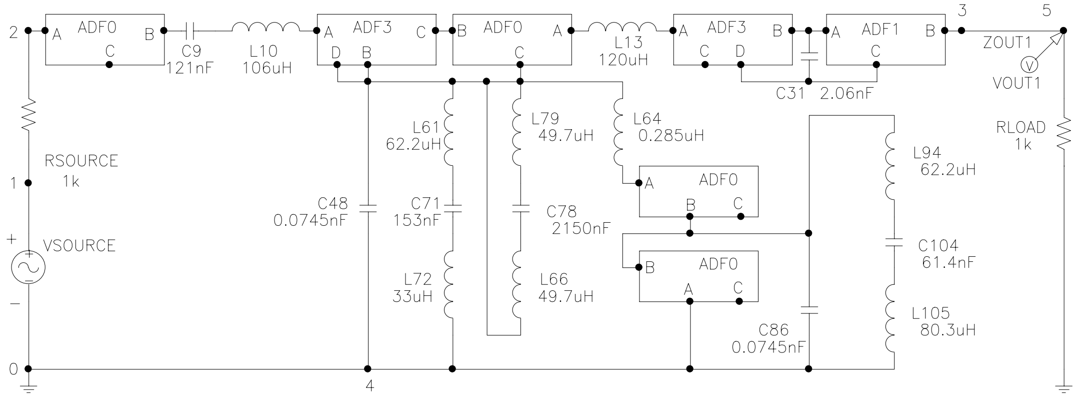
CIRCUIT-CONSTRUCTING PROGRAM TREE CONTAINING AUTOMATICALLY DEFINED FUNCTION ADF0 THAT DEVELOPS INTO THE TWO T-SECTIONS



DISCOVERING THE NUMBER OF SUBSTRUCTURES

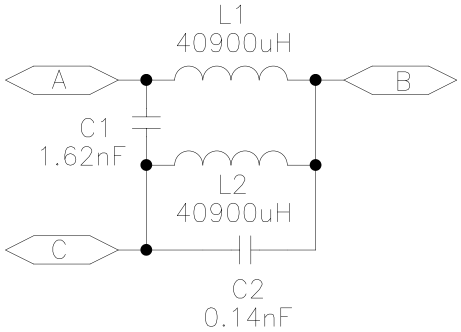
- **4 occurrences of 3-ported automatically defined function ADF0**
- **2 occurrences of 4-ported automatically defined function ADF3**
- **1 occurrence of 3-ported automatically defined function ADF1**

GENETICALLY EVOLVED DOUBLE-BANDPASS FILTER

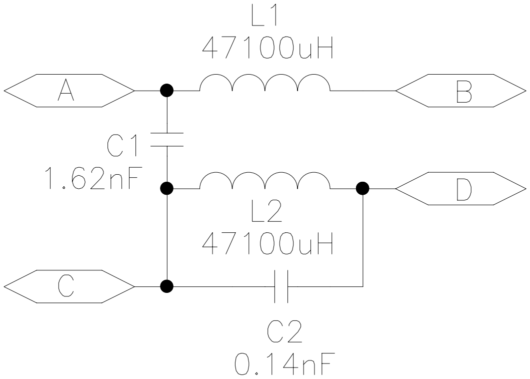


DISCOVERING THE NUMBER OF SUBSTRUCTURES

QUADRUPLY-USED SUBCIRCUIT PRODUCED BY AUTOMATICALLY DEFINED FUNCTION ADF0

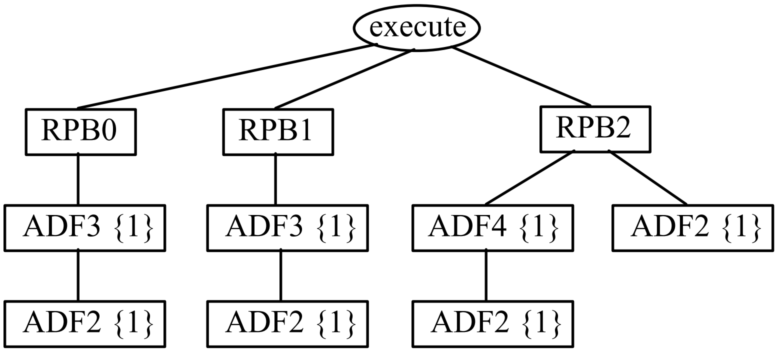


TWICE-USED FOUR-PORTED SUBCIRCUIT PRODUCED BY AUTOMATICALLY DEFINED FUNCTION ADF3

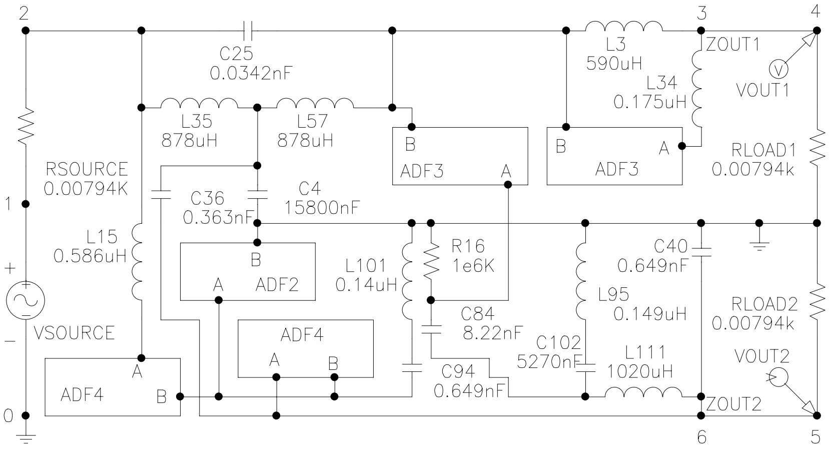


DISCOVERING THE NATURE OF THE HIERARCHICAL REFERENCES AMONG SUBSTRUCTURES

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



BEST-OF-RUN CIRCUIT FROM GENERATION 158



DISCOVERING THE NATURE OF THE HIERARCHICAL REFERENCES AMONG SUBSTRUCTURES

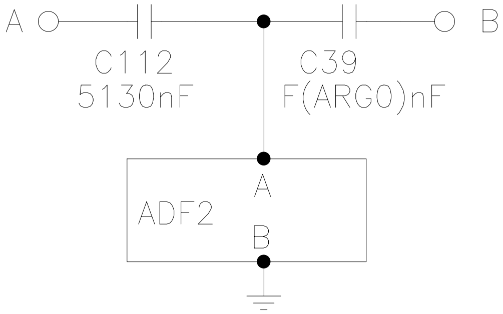
In addition to the mundane matter of inserting an ordinary 5,130-nanofarad capacitor C112, the execution of the three-ported automatically defined function ADF3

- **ADF3 inserts a *parameterized capacitor* C39 whose component value is dependent on the dummy variable ARG0 that is passed into automatically defined function ADF3 from the program that calls ADF3, and**
- **ADF3 calls automatically defined function ADF2. A dummy variable ARG0 is passed along from ADF3 to ADF2. In turn, ADF2 creates a *parameterized inductor* whose component value is dependent on this dummy variable.**

PASSING A PARAMETER TO A SUBSTRUCTURE

THREE-PORTED AUTOMATICALLY DEFINED FUNCTION ADF3 OF THE BEST-OF-RUN CIRCUIT FROM GENERATION 158 FOR GENETICALLY EVOLVED CROSSOVER (WOOFER-TWEETER) FILTER

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))))
```

DISCOVERING THE NUMBER OF ARGUMENTS POSSESSED BY SUBSTRUCTURES

- **In a run on the even-6-parity problem, the Genetic Programming Problem Solver (GPPS) determined that the genetically evolved solution would consist of**
  - **1 result-producing branch (RPB),**
  - **1 automatically defined loop (ADL), and**
  - **2 two-argument automatically defined functions.**
- **The architecture-altering operations in GPPS determined that the automatically defined functions would possess two arguments.**

A DEVELOPMENTAL PROCESS CAN PRESERVE LOCALITY

- **Because most of the component-creating, topology-modifying, and development-controlling functions operate on a small local area of the circuit, the subtrees that are transplanted by the crossover operation generally operate locally.**
- **Thus, when a crossover replaces a subtree in one individual with a subtree from another individual, it (usually) replaces a local structure in the circuit created by the first individual with a local structure in the circuit created by the second individual.**
- **The developmental process works in conjunction with the crossover operation in preserving locality.**

A DEVELOPMENTAL PROCESS CAN PRESERVE ELECTRICAL CONNECTIVITY

- There are no unconnected leads in the initial circuit.
- Each component-creating, topology-modifying, and development-controlling function preserves connectivity at each stage of the developmental process.
- The result is that there are no unconnected leads in the fully developed circuit.

A DEVELOPMENTAL PROCESS CAN FACILITATE REUSE

OFTEN-USED GA/ES REPRESENTATION FOR CIRCUIT SYNTHESIS

A mixture of real-valued variables, integer-valued variables, and categorical variables are encoded in the chromosome

L	.220	2	3	**C**	403.	3	6	**L**	.528	6	9	**L**	.041	9	0

- **Bit-string chromosome**

Resistor		**2.5 Ω**								**Node 3**			**Node 6**		
0	1	0	0	1	0	1	0	0	0	0	1	1	1	1	0

  - **The component type**
  - **The component value**
  - **The node to which the component's 1st lead is connected**
  - **The node to which the component's 2nd lead is connected**

PROBLEMS WITH OFTEN-USED GA/ES REPRESENTATION FOR CIRCUIT SYNTHESIS

Chromosome #1

1st Component | 2nd Component

L	.220	1	2	**C**	403.	2	0

Chromosome #2

1st Component | 2nd Component

R	250.	0	1	**C**	100.	1	2

Nominal Offspring #1 is invalid

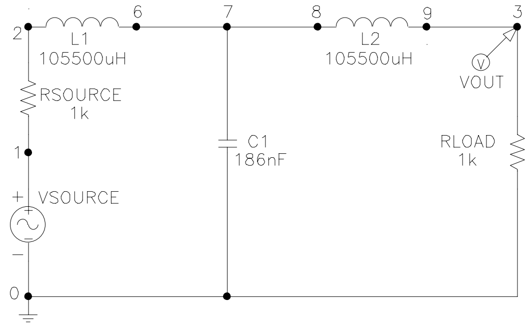
1st Component | 2nd Component

L	.220	1	2	**C**	100.	1	2

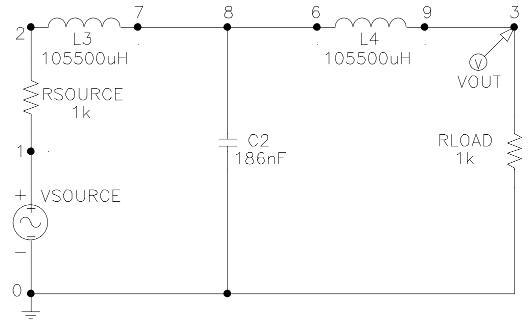
- **Penalize (in fitness measure)**
- **Delete**
- **Repair (most common method)**
- **Inundate**

PROBLEMS — CONTINUED

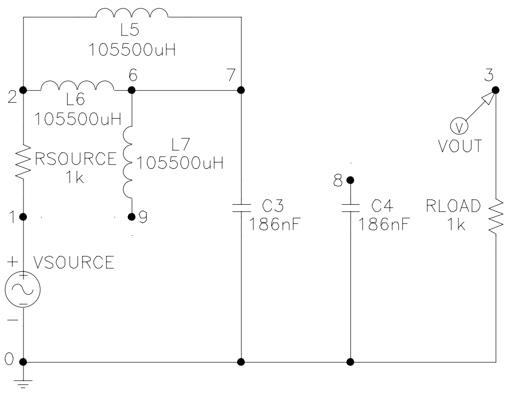
FIRST PARENT



SECOND PARENT



OFFSPRING



ADDITIONAL ISSUES

(6) discovering the type of substructures (e.g., subroutines, iterations, loops, recursions, or storage)
(7) discovering a general solution in the form of a parameterized topology containing free variables
(8) dynamically discovering the size and shape of the solution

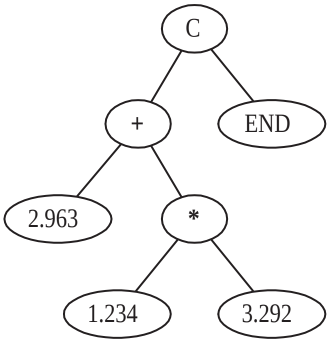
DISCOVERING THE TYPE OF SUBSTRUCTURES (E.G., SUBROUTINES, ITERATIONS, LOOPS, RECURSIONS, OR STORAGE)

- **The architecture-altering operations can dynamically create, duplicate, and delete subroutines, loops, recursions, and internal storage during a run**
- **In a run of the problem of evolving a computer program for the even-6-parity problem, the architecture-altering operations in GPPS determined that the program architecture would be**
  - **1 result-producing branch (RPB), and**
  - **2 automatically defined loops (ADLs).**

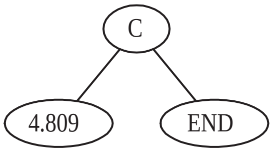
PARAMETERIZED TOPOLOGIES

VALUE-SETTING SUBTREES — 3 WAYS

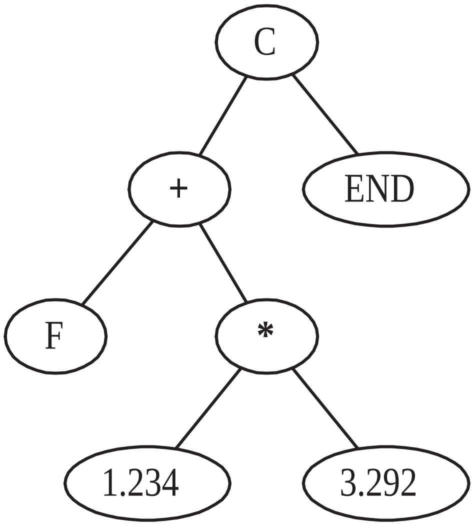
ARITHMETIC-PERFORMING SUBTREE



SINGLE PERTURBABLE CONSTANT



FREE VARIABLE F

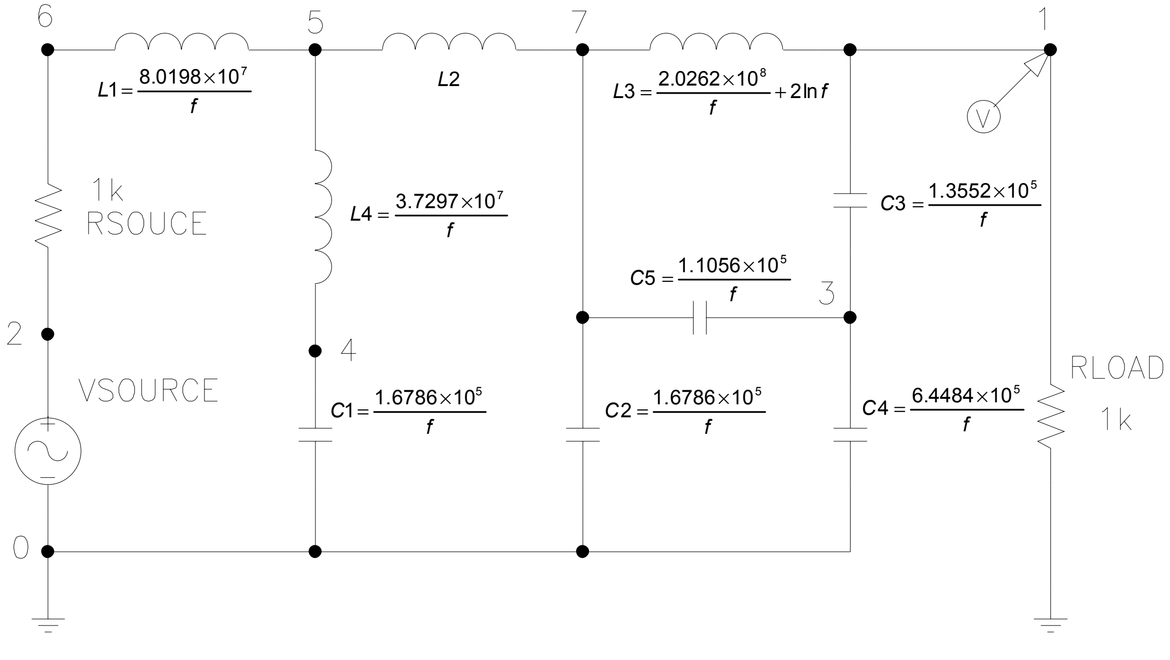


PARAMETERIZED TOPOLOGY FOR "GENERALIZED" LOWPASS FILTER

VARIABLE CUTOFF LOWPASS FILTER

- **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**

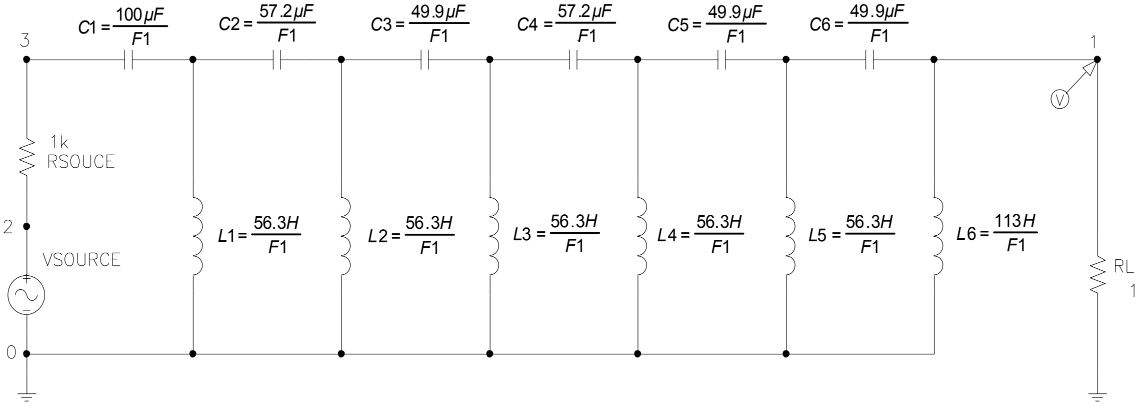
$$L2 = \frac{1.3406\times10^{-8}\left(4.7387\times10^{12}+f\right)\left(1.3331\times10^{16}+9.3714\times10^{5}f+f^{2}\right)}{f\left(3.4636\times10^{12}+f\right)}+\ln f \approx \frac{2.4451\times10^{8}}{f}+\ln f$$


6
5
7
1
$L1 = \frac{8.0198\times10^{7}}{f}$
$L2$
$L3 = \frac{2.0262\times10^{8}}{f}+2\ln f$
V
1k
RSOUCE
$L4 = \frac{3.7297\times10^{7}}{f}$
$C3 = \frac{1.3552\times10^{5}}{f}$
$C5 = \frac{1.1056\times10^{5}}{f}$
3
2
4
VSOURCE
RLOAD
$C1 = \frac{1.6786\times10^{5}}{f}$
$C2 = \frac{1.6786\times10^{5}}{f}$
$C4 = \frac{6.4484\times10^{5}}{f}$
1k
0

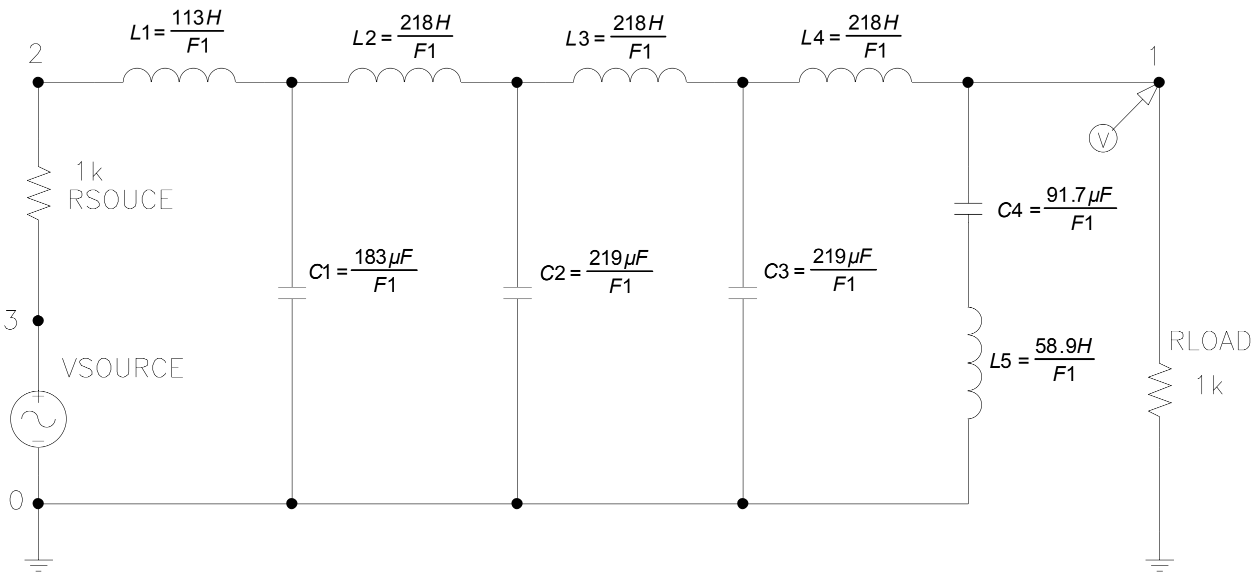
PARAMETERIZED TOPOLOGY USING CONDITIONAL DEVELOPMENTAL OPERATORS (GENETIC SWITCH)

VARIABLE-CUTOFF LOWPASS/HIGHPASS FILTER CIRCUIT

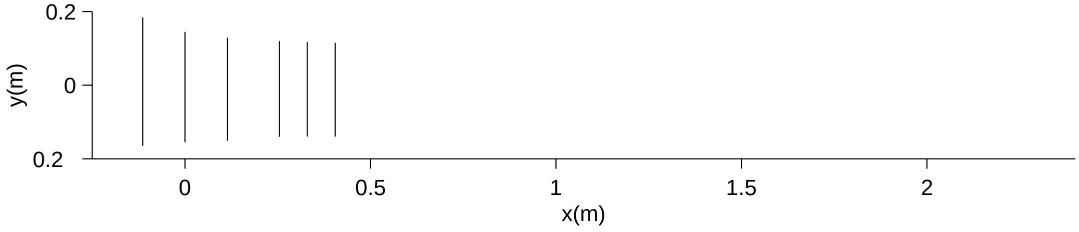
- **Best-of-run circuit from generation 93 when inputs call for a highpass filter (i.e., `F1` > `F2`)**



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


$L1 = \frac{113H}{F1}$
$L2 = \frac{218H}{F1}$
$L3 = \frac{218H}{F1}$
$L4 = \frac{218H}{F1}$
2
1
1k
RSOUCE
V
$C4 = \frac{91.7\mu F}{F1}$
$C1 = \frac{183\mu F}{F1}$
$C2 = \frac{219\mu F}{F1}$
$C3 = \frac{219\mu F}{F1}$
3
VSOURCE
$L5 = \frac{58.9H}{F1}$
RLOAD
1k
0

AUTOMATIC SYNTHESIS OF A YAGI-UDA WIRE ANTENNA USING GENETIC ALGORITHM (LINDEN 1997)



- **When the genetic algorithm (GA) operating on fixed-length character strings was used to synthesize a particular Yagi-Uda wire antenna by Linden (1997), the chromosome was based on**
  - **a particular number of reflectors (one) and**
  - **a particular number of directors.**

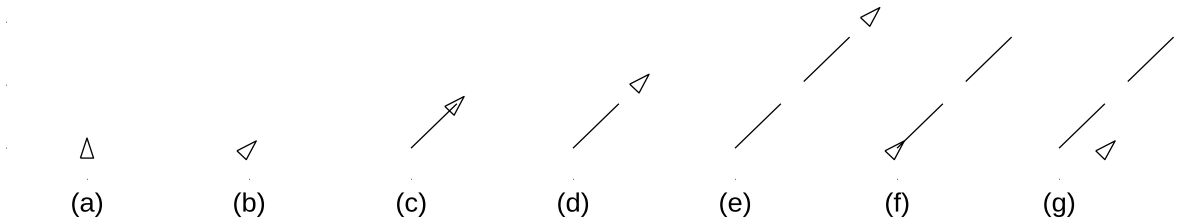
The chromosome encoded

- **the spacing between the parallel wires**
- **the length of each of the parallel wires**

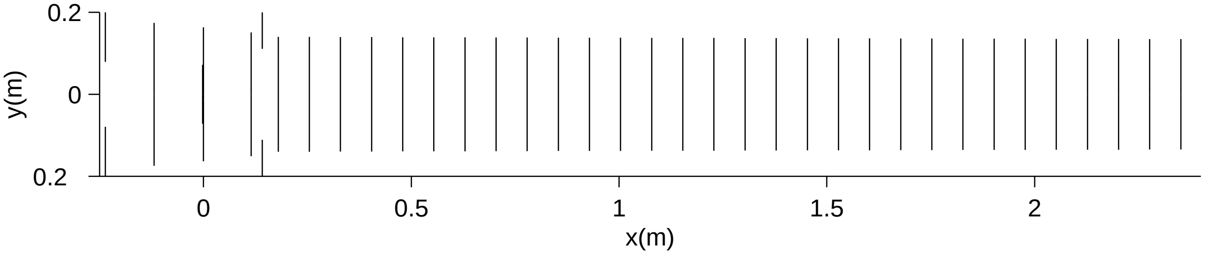
AUTOMATIC SYNTHESIS OF A WIRE ANTENNA

EXAMPLE OF TURTLE FUNCTIONS USED TO CREATE WIRE 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



- **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**