AUTOMATICALLY DEFINED FUNCTIONS (ADFs)

SUBROUTINES (PROCEDURES, SUBFUNCTIONS, DEFINED FUNCTION, DEFUN)

```
(PROGN (DEFUN exp (dv)
         (VALUES
           (+ 1.0
              dv
              (* 0.5 dv dv)
              (* 0.17 dv dv
dv)))

         (VALUE (+ (exp (* x x))
                   (exp (* 4 y))
                   (exp 2))))
```

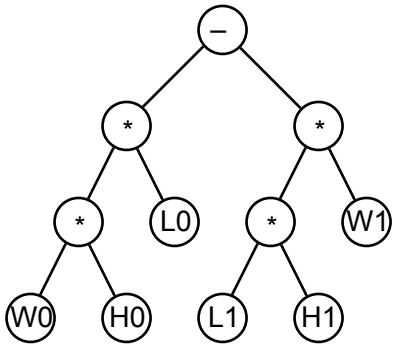
10 FITNESS-CASES SHOWING THE VALUE OF THE DEPENDENT VARIABLE, *D*, ASSOCIATED WITH THE VALUES OF THE SIX INDEPENDENT VARIABLES, L_0, W_0, H_0, L_1, W_1, H_1

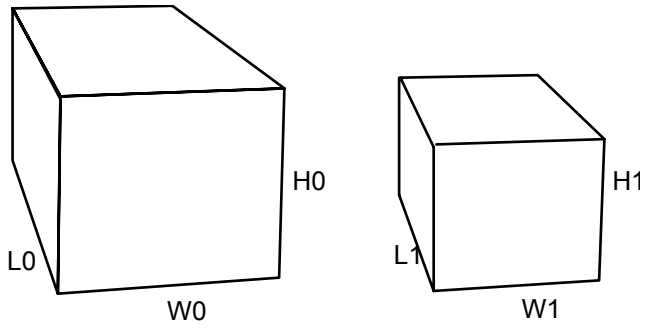
Fitness case	L_0	W_0	H_0	L_1	W_1	H_1	**Dependent variable *D***
1	3	4	7	2	5	3	**54**
2	7	10	9	10	3	1	**600**
3	10	9	4	8	1	6	**312**
4	3	9	5	1	6	4	**111**
5	4	3	2	7	6	1	**–18**
6	3	3	1	9	5	4	**–171**
7	5	9	9	1	7	6	**363**
8	1	2	9	3	9	2	**–36**
9	2	6	8	2	6	10	**–24**
10	8	1	10	7	5	1	**45**

SOLUTION WITHOUT AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

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

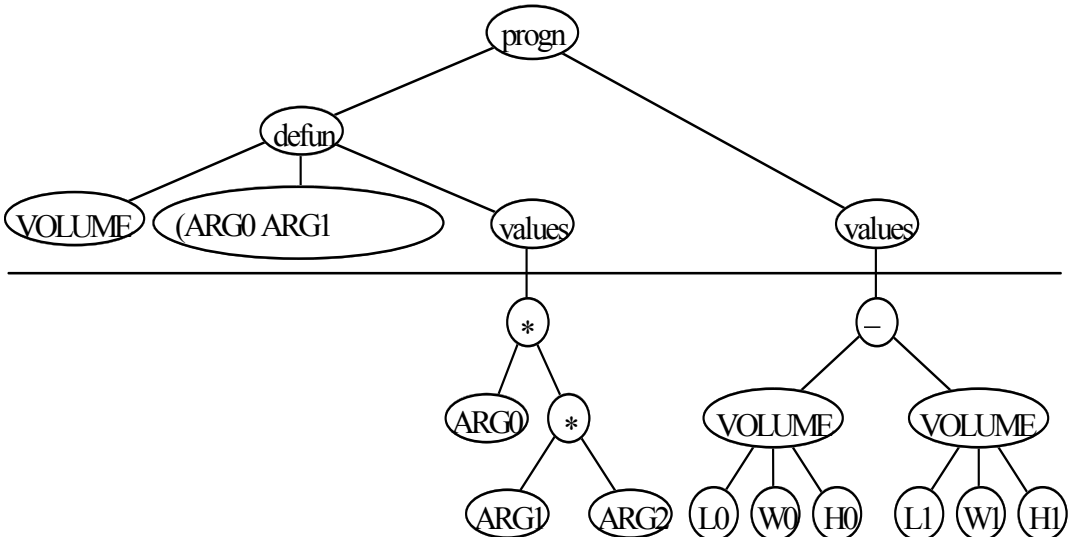
D = W0*L0*H0 – W1*L1*H1





AUTOMATICALLY DEFINED FUNCTIONS (SUBROUTINES - PROCEDURES - SUBFUNCTIONS - DEFUN'S)

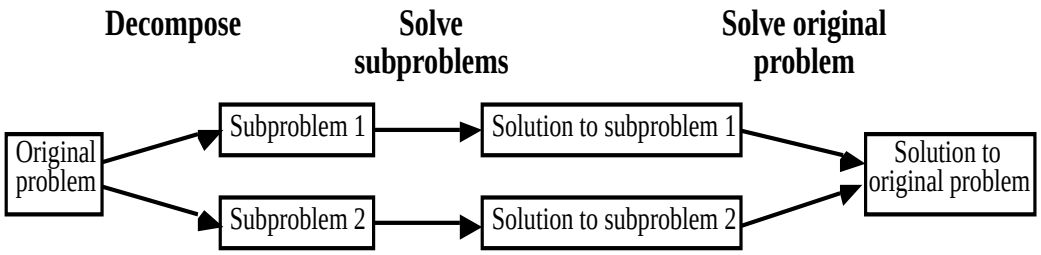
```
(progn
  (defun volume (arg0 arg1 arg2)
    (values
      (* arg0 (* arg1 arg2))))
(values  (- (volume L0 W0 H0)
            (volume L1 W1 H1))))
```



AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

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

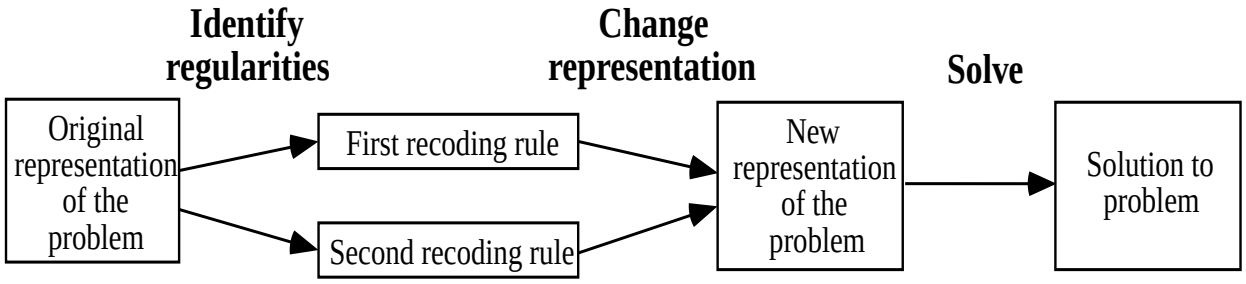
DIVIDE AND CONQUER



- **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 HEIRARCHICAL PROBLEM-SOLVING PROCESS



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

AFTER THE CHANGE OF REPRESENTATION, THERE ARE TWO NEW VARIABLES – V_0 AND V_1

THE 6-DIMENSIONAL NON-LINEAR REGRESSION PROBLEM BECOMES AN EASILY SOLVED 2-DIMENSIONAL LINEAR REGRESSION

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)

- **In generation 0, we create a population of programs, each consisting of a main result-producing branch (RPB) and one or more function-defining branches (automatically defined functions, ADFs, subroutines)**
  - **Different ingredients for RPB and ADFs**
  - **The terminal set of an ADF typically contains dummy arguments (formal parameters), such as `ARG0`, `ARG1`, …**
  - **The function set of the RPB contains `ADF0`, …**
  - **ADFs are private and associated with a particular individual program in the population**
- **The entire program is executed and evaluated for fitness**

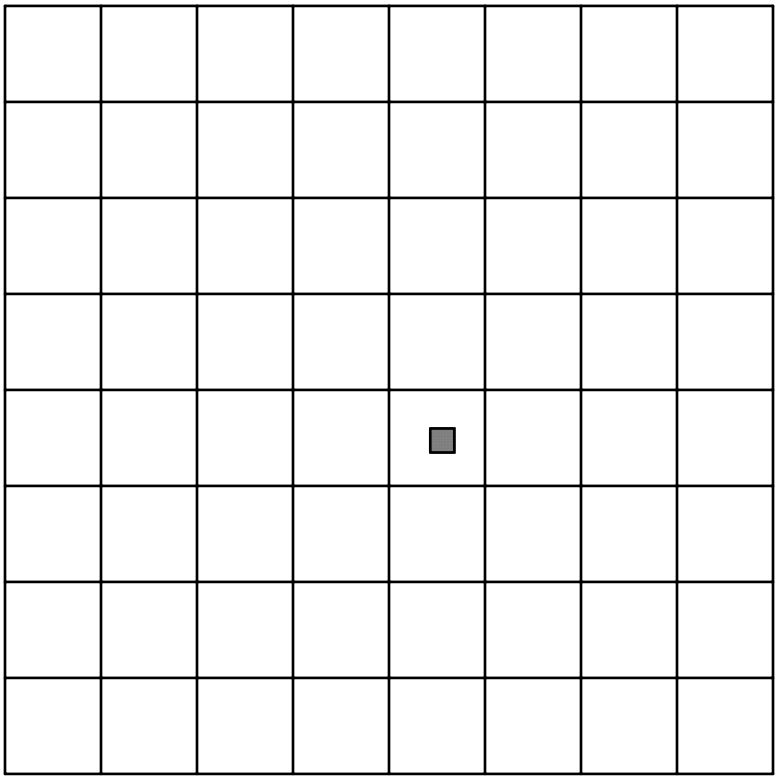
AUTOMATICALLY DEFINED FUNCTIONS (ADFs, SUBROUTINES)

- **Genetic operation of reproduction is the same as before**
- **Mutation operation starts (as before) by picking a mutation point from either RPB or an ADF and deleting the subtree rooted at that point. As before, a subtree is then grown at the point. The new subtree is composed of the allowable ingredients for that point — so that the result is a syntactically valid executable program.**
- **Crossover operation starts (as before) by picking a crossover point from either RPB or an ADF of one parent. The choice of crossover point in the second parent is then restricted (e.g., to the RPB or to the ADF) — so that when the subtrees are swapped, the result is a syntactically valid executable program.**

MAIN POINTS FROM GP-2 BOOK (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.**

LAWN MOWER PROBLEM
(LAWN SIZE 64)



GP TABLEAU WITHOUT ADF'S FOR THE 64-SQUARE LAWNMOWER

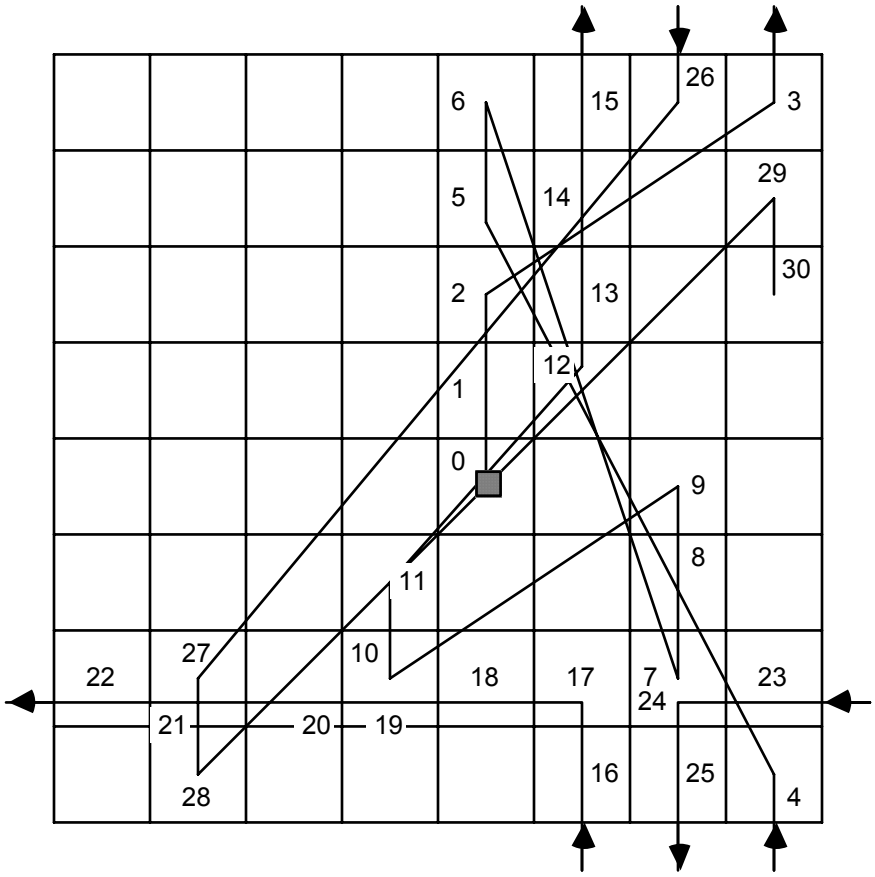
Objective:	**Find a program to control a lawnmower so that it mows all 64 squares of grass in an unobstructed toroidal yard.**
Terminal set without ADFs:	**(LEFT), (MOW), and the random constants $\Re_{v8}$.**
Function set without ADFs:	**V8A, FROG, and PROGN.**
Fitness cases:	**One fitness case consisting of a toroidal lawn with 64 squares, each initially containing grass.**
Raw fitness:	**Amount of grass (from 0 to 64) mowed within the maximum allowed number of state-changing operations.**
Standardized fitness:	**Total number of squares (i.e., 64) minus raw fitness.**
Hits:	**Same as raw fitness.**
Wrapper:	**None.**
Parameters:	**M = 1,000. G = 51.**

Success predicate:	**A program scores the maximum number of hits.**

296-POINT SOLUTION FROM GENERATION 34 WITHOUT ADF'S – LAWN SIZE 64

```
(V8A (V8A (V8A (FROG (PROGN (PROGN (V8A (MOW) (MOW)) (FROG #(3
2))) (PROGN (V8A (PROGN (V8A (PROGN (PROGN (MOW) #(2 4)) (FROG
#(5 6))) (PROGN (V8A (MOW) #(6 0)) (FROG #(2 2)))) (V8A (MOW)
(MOW))) (PROGN (V8A (PROGN (PROGN #(0 3) #(7 2)) (FROG #(5
6))) (PROGN (V8A (MOW) #(6 0)) (FROG #(2 2)))) (V8A (MOW)
(MOW)))) (PROGN (FROG (MOW)) (PROGN (PROGN (PROGN (V8A (MOW)
(MOW)) (FROG (LEFT))) (PROGN (MOW) (V8A (MOW) (MOW)))) (PROGN
(V8A (PROGN #(0 3) #(7 2)) (V8A (MOW) (MOW))) (PROGN (V8A
(MOW) (MOW)) (PROGN (LEFT) (MOW)))))))) (V8A (PROGN (V8A
(PROGN (PROGN (MOW) #(2 4)) (FROG #(5 6))) (PROGN (V8A (MOW)
#(6 0)) (FROG #(2 2)))) (V8A (MOW) (MOW))) (V8A (FROG (LEFT))
(FROG (MOW))))) (V8A (FROG (V8A (PROGN (V8A (PROGN (V8A (MOW)
(MOW)) (FROG #(3 7))) (V8A (PROGN (MOW) (LEFT)) (V8A (MOW) #(5
3)))) (PROGN (PROGN (V8A (PROGN (LEFT) (MOW)) (V8A #(1 4)
(LEFT))) (PROGN (FROG (MOW)) (V8A (MOW) #(3 7)))) (V8A (PROGN
(FROG (MOW)) (V8A (LEFT) (MOW))) (V8A (FROG #(1 2)) (V8A (MOW)
(LEFT)))))) (PROGN (V8A (FROG #(3 1)) (V8A (FROG (PROGN (PROGN
(V8A (MOW) (MOW)) (FROG #(3 2))) (FROG (FROG #(5 0))))) (V8A
(PROGN (FROG (MOW)) (V8A (MOW) (MOW))) (V8A (FROG (LEFT))
(FROG (MOW)))))) (PROGN (PROGN (PROGN (PROGN (LEFT) (MOW))
(V8A (MOW) #(3 7))) (V8A (V8A (MOW) (MOW)) (PROGN (LEFT)
(LEFT)))) (V8A (FROG (PROGN #(3 0) (LEFT))) (V8A (PROGN (MOW)
(LEFT)) (FROG #(5 4)))))))) (PROGN (FROG (V8A (PROGN (V8A
(PROGN (PROGN (V8A (PROGN (PROGN (MOW) #(2 4)) (FROG #(5 6)))
(PROGN (V8A (MOW) #(1 2)) (FROG #(2 2)))) (V8A (MOW) (MOW)))
(FROG #(3 7))) (V8A (PROGN (PROGN (MOW) #(2 4)) (FROG #(5 6)))
(PROGN (V8A (MOW) #(6 0)) (FROG #(2 2))))) (PROGN (PROGN (V8A
(FROG (MOW)) (V8A #(1 4) (LEFT))) (PROGN (FROG (MOW)) (V8A
(MOW) #(3 7)))) (V8A (PROGN (FROG (MOW)) (V8A (LEFT) (MOW)))
(V8A (FROG #(1 2)) (V8A (MOW) (LEFT)))))) (PROGN (V8A (PROGN
(FROG #(2 4)) (V8A (MOW) (MOW))) (V8A (FROG (MOW)) (LEFT)))
(PROGN #(3 0) (LEFT))))) (FROG (V8A #(7 4) (MOW)))))) (V8A
(V8A (PROGN (MOW) #(4 3)) (V8A (LEFT) #(6 1))) (MOW)))
```

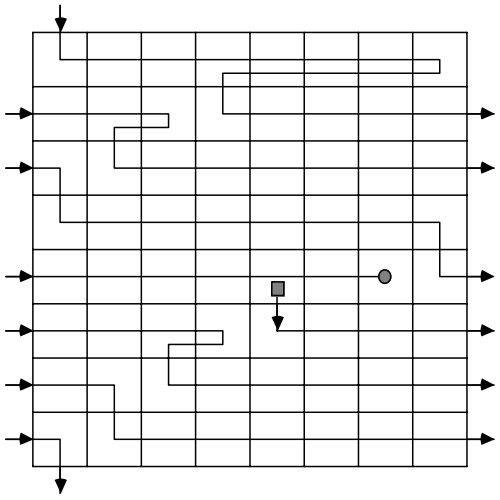
PARTIAL TRAJECTORY OF 296-POINT BEST-OF-RUN PROGRAM FROM GENERATION 34 FOR MOWING OPERATIONS 0 THROUGH 30 WITHOUT ADF'S OF WITHOUT ADF'S – LAWN SIZE 64



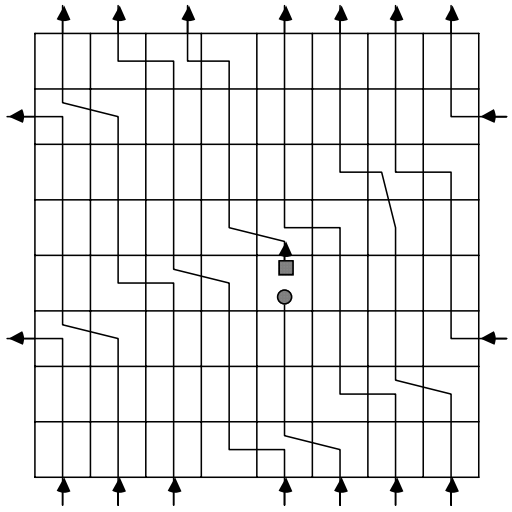
AVERGAGE-SIZED (78-POINT) SOLUTION WITH ADF'S – LAWN SIZE 64 (16-WAY DECOMPOSITION WITH HIERARCHICAL CALLS)

```
(progn (defun ADF0 ()

(values (V8A (PROGN (V8A (V8A (LEFT) #(6 5)) (PROGN
(MOW) (LEFT))) (V8A (PROGN (MOW) (MOW)) (V8A (MOW)
(MOW)))) (V8A (PROGN (V8A #(1 4) (MOW)) (PROGN #(3 1)
(MOW))) (PROGN (PROGN #(3 1) (MOW)) (PROGN (LEFT)
(LEFT)))))))

      (defun ADF1 (ARG0)

(values (V8A (PROGN (FROG (PROGN ARG0 (ADF0))) (V8A
(PROGN (MOW) (ADF0)) (V8A (V8A (ADF0) #(3 4)) (V8A
(ADF0) ARG0)))) (V8A (FROG (FROG (MOW))) (PROGN
(PROGN (MOW) #(3 5)) (PROGN (MOW) (MOW)))))))

      (values (V8A (ADF1 (ADF1 (V8A #(7 1) (LEFT))))
(V8A (V8A (PROGN (LEFT) (LEFT)) (V8A #(7 0) (LEFT)))
(FROG (V8A (ADF0) (MOW)))))))
```

TRAJECTORY OF AVERGAGE-SIZED (78-POINT) SOLUTION WITH ADF'S (8-WAY DECOMPOSITION)



TRAJECTORY OF 42-POINT SOLUTION (16-WAY DECOMPOSITION) (GEN 5)

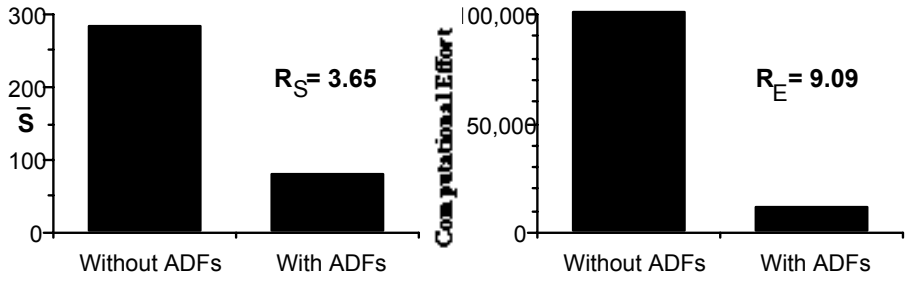


TYPES OF TRAJECTORIES IN LAWN-MOWER PROBLEM

Category	Percentage of runs
Row or column moving	49%
Zigzagging	20%
Large swirls	17%
Crisscrossing	10%
Tight swirls	4%

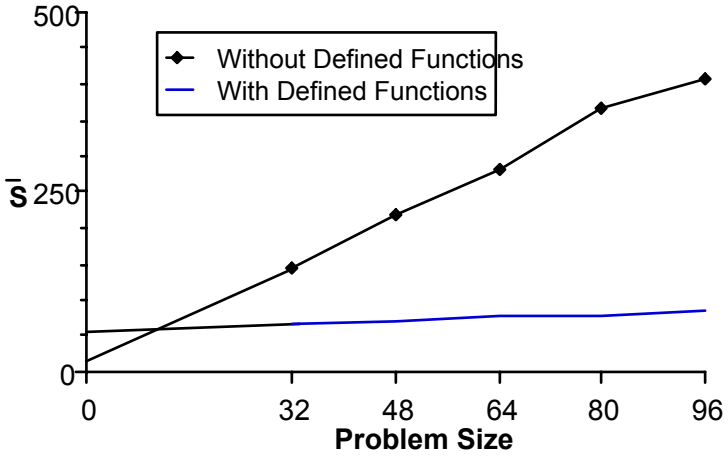
COMPARISON TABLE FOR THE LAWN MOWER PROBLEM – LAWN SIZE 64

	Without ADF'S	With ADF'S
Average Structural Complexity $\bar{s}$	**280.82**	**76.95**
Computational Effort ***I(M,i,z)***	**100,000**	**11,000**



COMPARISON OF AVERAGE STRUCTURAL COMPLEXITY FOR LAWN SIZES OF 32, 48, 64, 80, AND 96 WITH AND WITHOUT ADF'S

	32	48	64	80	96
$\bar{S}_{without}$	145.0	217.6	280.8	366.1	408.8
$\bar{S}_{with}$	66.3	69.0	76.9	78.8	84.9



WITHOUT ADF'S

$S = 13.2 + 4.2L$

Correlation R of 1.00

WITH ADF'S

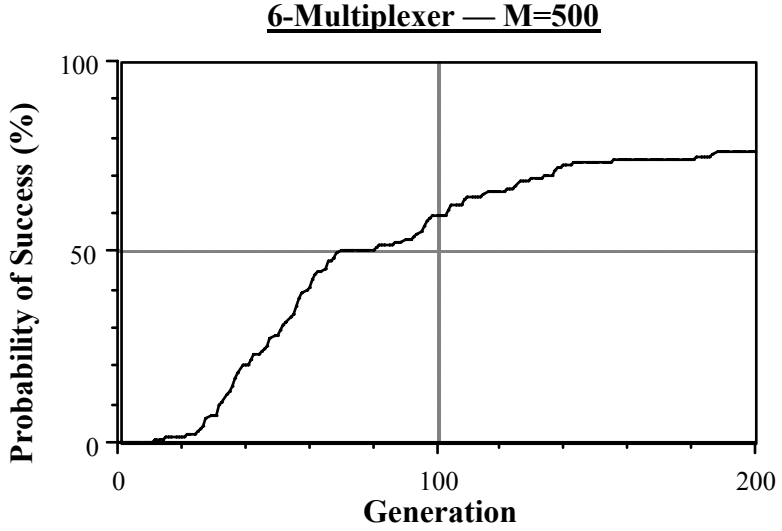
$S = 56.4 + 0.29L$

Correlation R of 0.99

MEASURING COMPUTATIONAL EFFORT *E*

COMPUTATIONAL EFFORT

CUMULATIVE PROBABILITY OF SUCCESS *P*(*M*,*I*) FOR THE 6-MULTIPLEXER PROBLEM WITH A POPULATION SIZE *M* = 500 FOR GENERATIONS 0 THROUGH 200



NUMBER OF INDEPENDENT RUNS *R*(*Z*) REQUIRED AS A FUNCTION OF THE CUMULATIVE PROBABILITY

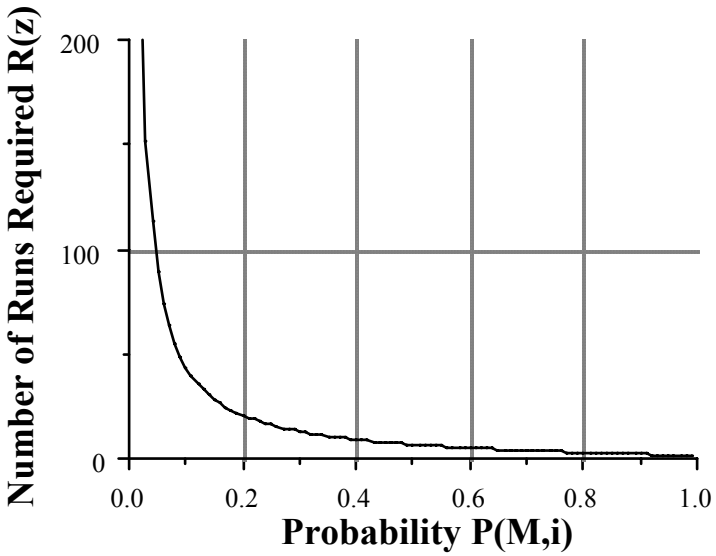
$$z = 1 - [1 - P(M,i)]^R$$

$$R(z) = \left\lceil \frac{\log(1-z)}{\log(1-P(M,i))} \right\rceil$$

NUMBER OF INDEPENDENT RUNS *R*(*Z*) REQUIRED AS A FUNCTION OF THE CUMULATIVE PROBABILITY OF SUCCESS *P(M,I)* FOR *Z* = 99%

$$z = 1 - [1 - P(M,i)]^R$$

$$R(z) = \left\lceil \frac{\log(1-z)}{\log(1-P(M,i))} \right\rceil$$



TOTAL NUMBER OF INDIVIDUALS THAT MUST BE PROCESSED FOR THE 6-MULTIPLEXER PROBLEM WITH A POPULATION SIZE $M = 500$

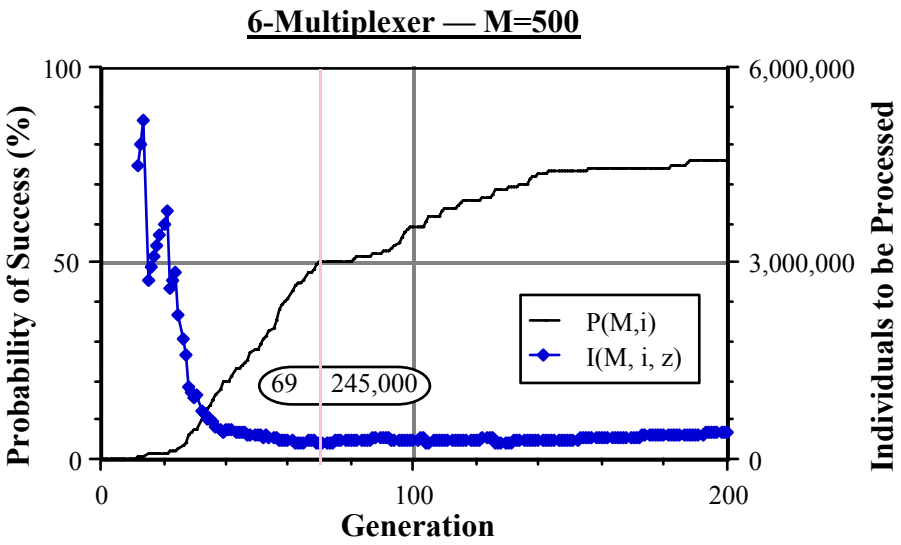
Gen	Cumulative probability of success $P(M,i)$	Number of independent runs $R(z)$ required	Total number of individuals that must be processed $I(M,i,z)$
25	3%	152	1,976,000
50	28%	15	382,500
100	59%	6	303,000
150	73%	4	302,000
200	76%	4	402,000

TOTAL NUMBER OF INDIVIDUALS THAT MUST BE PROCESSED FOR THE 6-MULTIPLEXER PROBLEM WITH A POPULATION SIZE $M = 500$

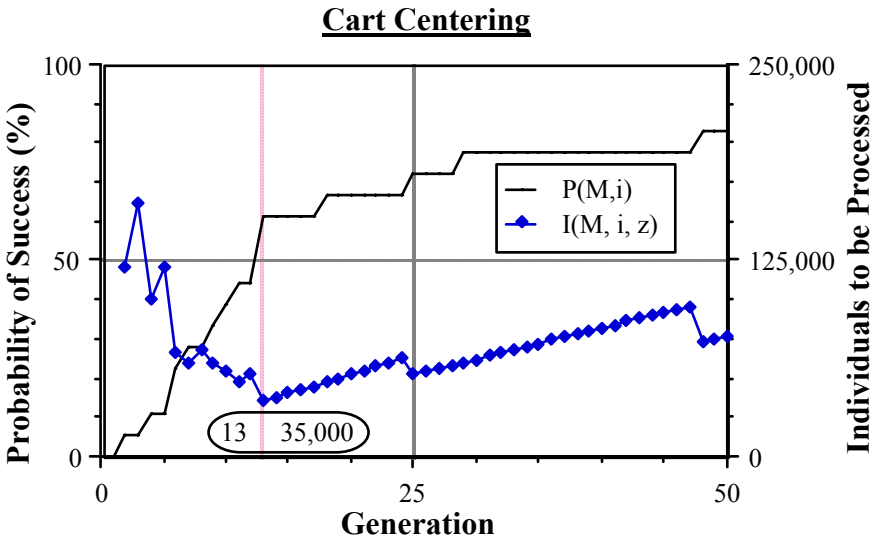
- **Make multiple runs of problem**
- **Experimentally observe $P(M,i)$ for each i**
- **Probability $z = 0.99$**
- $$R(z) = \frac{\log (1-z)}{\log(1-P(M,i))}$$
- $I(M,i,z) = M (i+1) R(z)$
- **Best generation i* minimizes $I(M,i,z)$**
- **Computational effort $E = I(M,i^*,z)$**

$= M (i^*+1) R(z)$

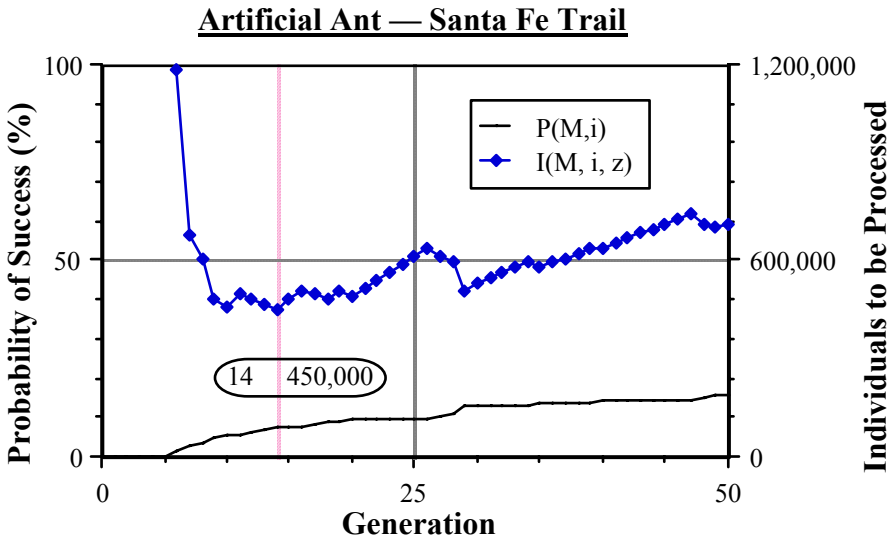
PERFORMANCE CURVES FOR THE 6-MULTIPLEXER PROBLEM WITH A POPULATION SIZE $M = 500$ FOR GENERATIONS 0 THROUGH 200



PERFORMANCE CURVES FOR POPULATION SIZE $M = 500$ FOR THE CART CENTERING PROBLEM

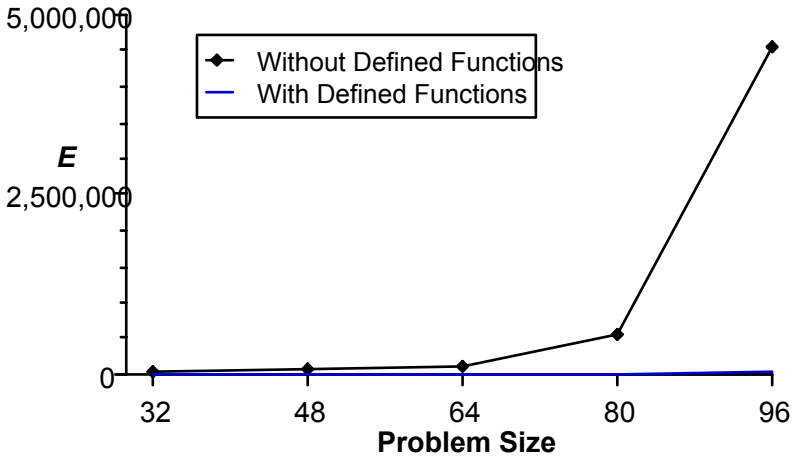


PERFORMANCE CURVES FOR POPULATION SIZE $M = 500$ FOR THE ARTIFICIAL ANT PROBLEM WITH THE SANTA FE TRAIL



COMPARISON OF COMPUTATIONAL EFFORT FOR LAWN SIZES OF 32, 48, 64, 80, AND 96 WITH AND WITHOUT ADF'S

	32	48	64	80	96
$E_{without}$	19,000	56,000	100,000	561,000	4,692,000
E_{with}	5,000	9,000	11,000	17,000	20,000



WITHOUT ADF'S

$E = -2{,}855{,}000 + 61{,}570L$

Correlation R of 0.77

$E = 944.2 * 10^{0.362 L}$

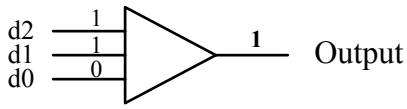
Correlation R of 0.98

WITH ADF'S

$E = -2{,}800 + 2.37L$

Correlation R of 0.99

BOOLEAN EVEN-3-PARITY FUNCTION



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

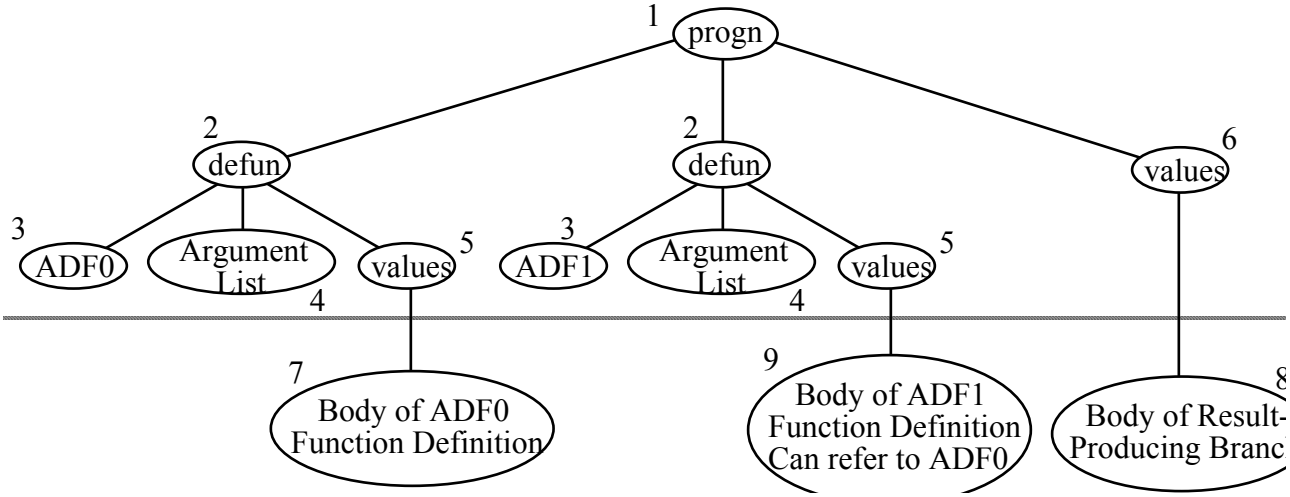
GP TABLEAU WITHOUT ADFS FOR THE EVEN-3-PARITY PROBLEM

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

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

HIERARCHICAL AUTOMATICALLY DEFINED FUNCTIONS

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



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

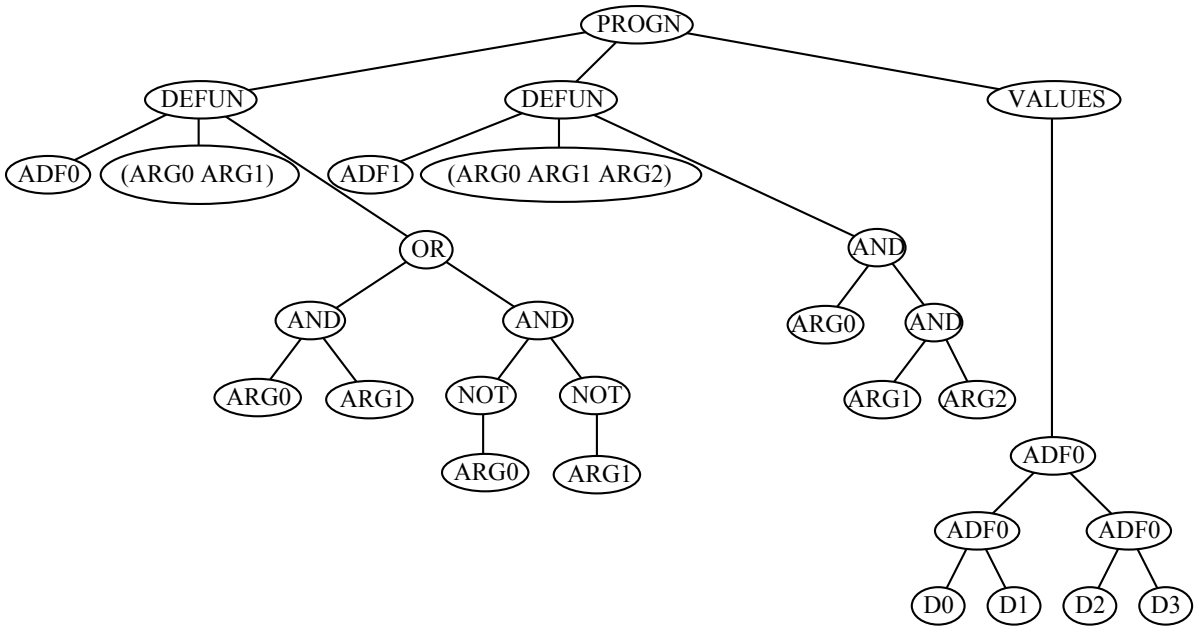
Objective:	**Find a program that produces the value of the Boolean even-3-parity function as its output when given the value of the three independent variables as its input.**
Architecture of the overall program with ADFs:	**One result-producing branch and two two-argument function-defining branches,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

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

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

- **ADF0 is even-2-parity (equivalence)**
- **RPB calls on ADF0 3 times**

  (ADF0 (ADF0 D0 D1) (ADF0 D2 D3))
- **ADF1 is ignored by RPB**



EVEN-4-PARITY WITH ADF'S

- **GEN 12 - 74 POINTS - 16 HITS (OUT OF 16)**

```
(PROGN  (DEFUN ADF0 (ARG0 ARG1)
        (VALUES(NAND (OR (AND (NOR ARG0
ARG1) (NOR (AND ARG1 ARG1) ARG1)) (NOR
(NAND ARG0 ARG0) (NAND ARG1 ARG1))) (NAND
(NOR (NOR ARG1 ARG1) (AND (OR (NAND ARG0
ARG0) (NOR ARG1 ARG0)) ARG0)) (AND (OR
ARG0 ARG0) (NOR (OR (AND (NOR ARG0 ARG1)
(NAND ARG1 ARG1)) (NOR (NAND ARG0 ARG0)
(NAND ARG1 ARG1))) ARG1))))))

        (DEFUN ADF1 (ARG0 ARG1 ARG2)
        (VALUES (OR (AND ARG2 (NAND ARG0
ARG2)) (NOR ARG1 ARG1)))

        (VALUES
        (ADF0 (ADF0 D0 D2) (NAND (OR D3
D1) (NAND D1 D3))))
```

- **ADF0 is XOR. ADF1 is not called.**
- **RPB simplifies to**

  (XOR (XOR D0 D2) (EQV D3 D1))

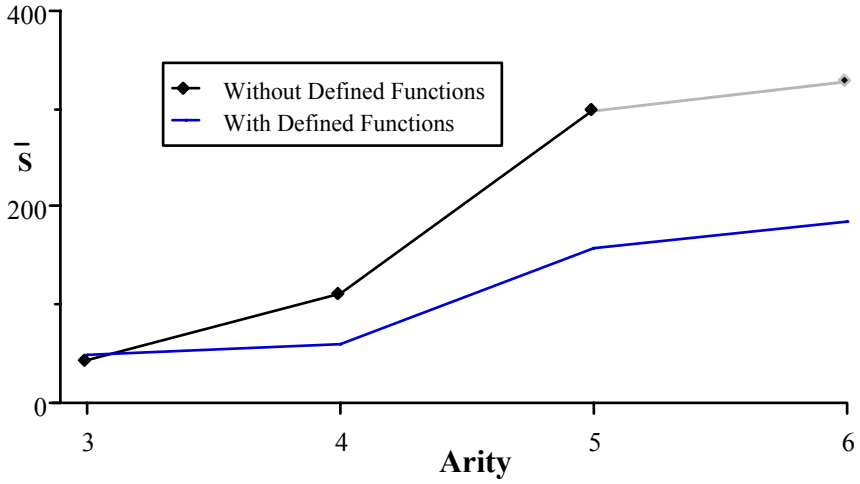
- **ADF1 is ignored**

SUMMARY OF THE STRUCTURAL COMPLEXITY RATIO R_S AND THE EFFICIENCY RATIO R_E FOR THE EVEN-PARITY PROBLEM OF ORDERS 3, 4, 5, AND 6

Problem	Structural complexity ratio R_S	Efficiency ratio R_E
Even-3-parity	0.92	1.50
Even-4-parity	1.87	2.18
Even-5-parity	1.91	14.07
Even-6-parity	1.77	52.2

COMPARISON OF AVERAGE STRUCTURAL COMPLEXITY OF SOLUTIONS TO EVEN-PARITY PROBLEM OF ORDERS 3, 4, 5, AND 6 WITH AND WITHOUT ADF'S

	3	4	5	6
$\bar{S}_{without}$	44.6	112.6	299.9	328.0
$\bar{S}_{with}$	48.2	60.1	156.8	184.8



WITHOUT ADF'S

$\bar{S}_{without} = -270.6 + 103.8A$

Correlation of 0.96

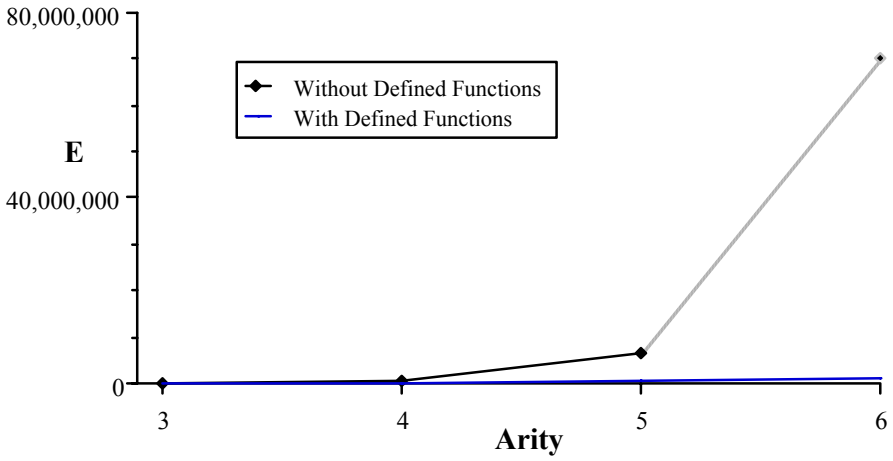
WITH ADF'S

$\bar{S}_{with} = -115.5 + 50.6A$

Correlation of 0.95

COMPARISON OF COMPUTATIONAL EFFORT FOR EVEN-PARITY PROBLEM WITH AND WITHOUT ADF'S

	3	4	5	6
$E_{without}$	96,000	384,000	6,528,000	70,176,000
E_{with}	64,000	176,000	464,000	1,344,000



WITHOUT ADF'S

$E_{without}$ = **78,100,000 = 21,640,000 A**

Correlation of 0.82

$E_{without}$ = **77.1 * 10 0.982 *A***

Correlation of 0.99

WITH ADF'S

E_{with} = **–1,350,000 + 413,000 *A***

Correlation of 0.92

E_{with} = **3070 * 10 0.439 *A***

Correlation of 0.99

TYPES OF SOLUTIONS TO THE EVEN-5-PARITY PROBLEM

Category	Percentage of runs
Lower-order parity functions in both `ADF0` and `ADF1`	5%
A lower-order parity function in either `ADF0` or `ADF1`, but not both	37%
No lower-order parity function in either `ADF0` or `ADF1`.	58%

TYPES OF EVEN-5-PARITY SOLUTIONS

	ADF0	Parity rule?	ADF1	Parity rule?
1	23130	Yes	15555	Yes
2	01285	No	15420	Yes
3	03920	No	13260	Yes
4	61455	Yes	21845	No
5	13260	Yes	65535	No
6	04010	No	21930	Yes
7	50115	Yes	13226	No
8	50115	Yes	13226	No
9	07420	No	13159	No
10	42469	No	19568	No
11	43600	No	52392	No
12	61680	No	43690	No
13	25198	No	59135	No
14	29199	No	02176	No
15	14192	No	65535	No
16	64201	No	58431	No
17	45067	No	63487	No
18	40960	No	53232	No
19	00596	No	27560	No

EVEN-11-PARITY WITH ADF'S

- **GENERATION 21 - 220 POINTS - 2,048 HITS (OUT OF 2,048)**
- **ADF0 is (EVEN-2-PARITY ARG1 ARG2)**
- **ADF1 is (EVEN-4-PARITY ARG0 ARG1 ARG2 ARG3)**

```
(PROGN

      (DEFUN ADF0 (ARG0 ARG1)

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

      (DEFUN ADF1 (ARG0 ARG1 ARG2)

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

      (VALUES

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

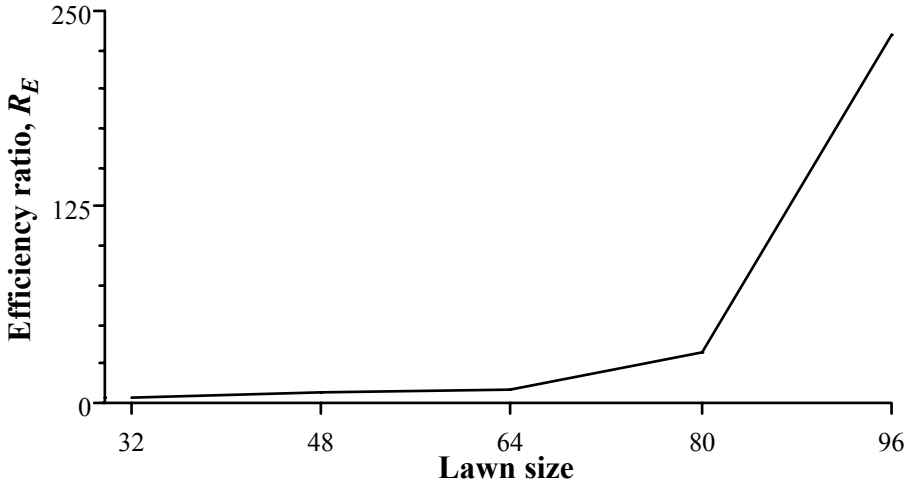
EVEN-11-PARITY WITH ADF'S

- **GENERATION 21 - 2,048 HITS (OUT OF 2,048) – SIMPLIFIED**

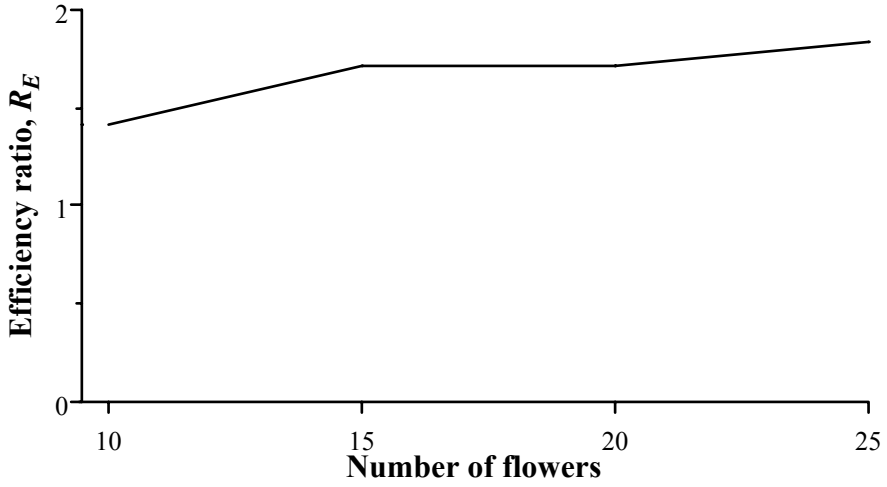
```
(OR (EVEN-4-PARITY D1 D0 (EVEN-
2-PARITY (EVEN-4-PARITY (EVEN-
2-PARITY D3 D2) (OR D10 D1)
(EVEN-4-PARITY D3 D4 D6 D7)
(EVEN-2-PARITY D8 D9)) (EVEN-2-
PARITY (NAND D1 D10) (EVEN-2-
PARITY D5 D3))) D1) (NOR (NAND
D1 D10) (EVEN-2-PARITY (NOR D10
D6) (NOR D1 D2))))
```

EFFICIENCY-RATIO SCALING FOR THE EVEN-PARITY PROBLEMS

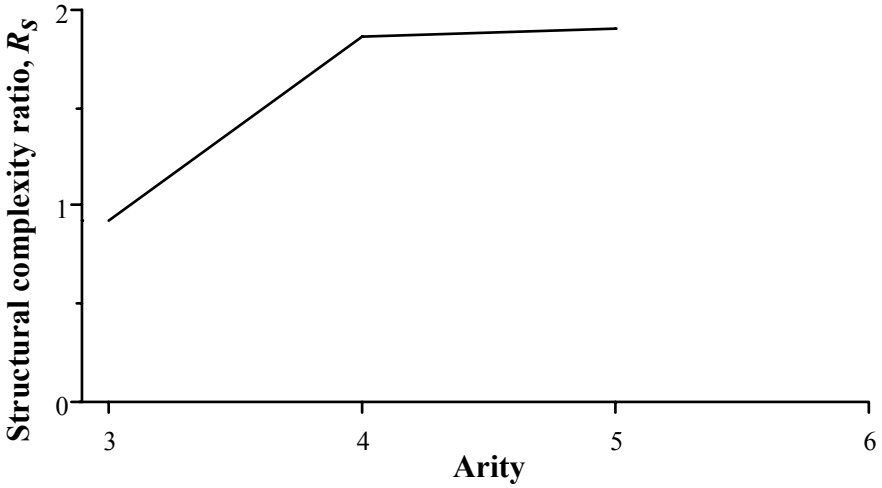
EFFICIENCY-RATIO SCALING FOR THE LAWNMOWER PROBLEM WITH A LAWN SIZE OF 32, 48, 64, 80, AND 96



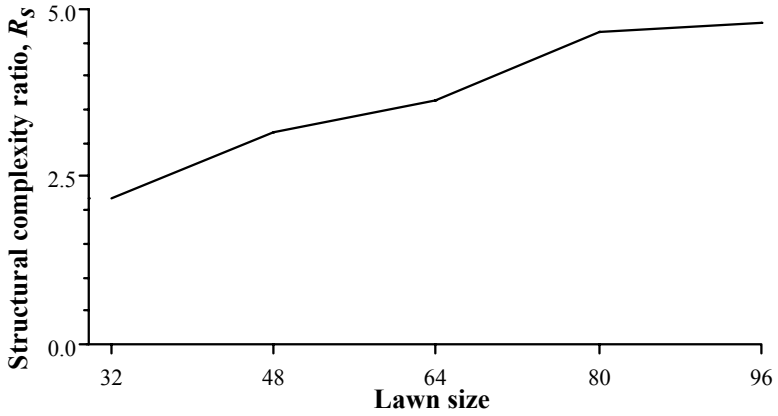
EFFICIENCY-RATIO SCALING FOR THE BUMBLEBEE PROBLEM WITH 10, 15, 20, AND 25 FLOWERS



GRAPH OF STRUCTURAL-COMPLEXITY-RATIO SCALING FOR THE EVEN-PARITY PROBLEMS



STRUCTURAL-COMPLEXITY-RATIO SCALING FOR THE LAWNMOWER PROBLEM WITH A LAWN SIZE OF 32, 48, 64, 80, AND 96



STRUCTURAL-COMPLEXITY-RATIO SCALING FOR THE BUMBLEBEE PROBLEM WITH 10, 15, 20, AND 25 FLOWERS