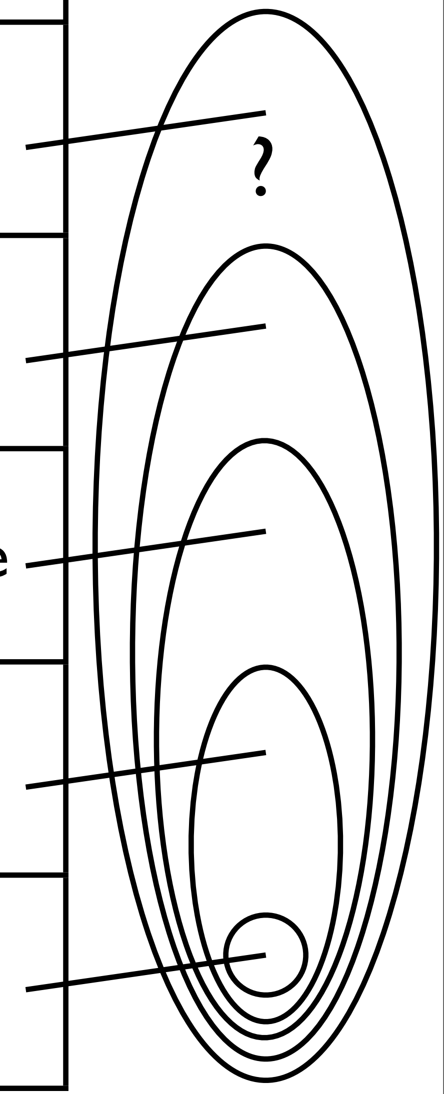
Genomes as Reactive Systems: A Computational Perspective

Lee Spector
Cognitive Science
Hampshire College

Outline

- Finite automata (with self regulation)
- Evolutionary computing
- Genomes = computer programs
- Environmentally mediated expression

Computers	Rules	Languages
Quantum Computers	Schrödinger's Equation	?
Turing Machines	$\alpha \rightarrow \beta$, unrestricted	Recursively Enumerable
Linear Bounded Automata	$\alpha \rightarrow \beta,\ \lvert\beta\rvert \geq \lvert\alpha\rvert$	Context-Sensitive
Pushdown Automata	$A \rightarrow \alpha, \alpha \in (V \cup T)*$	Context-Free
Finite Automata	$A \rightarrow wB, A \rightarrow w$	Regular



Human Finite Automaton

- Random word & target

Human Finite Automaton

- Random word & target
- Red & blue words & targets

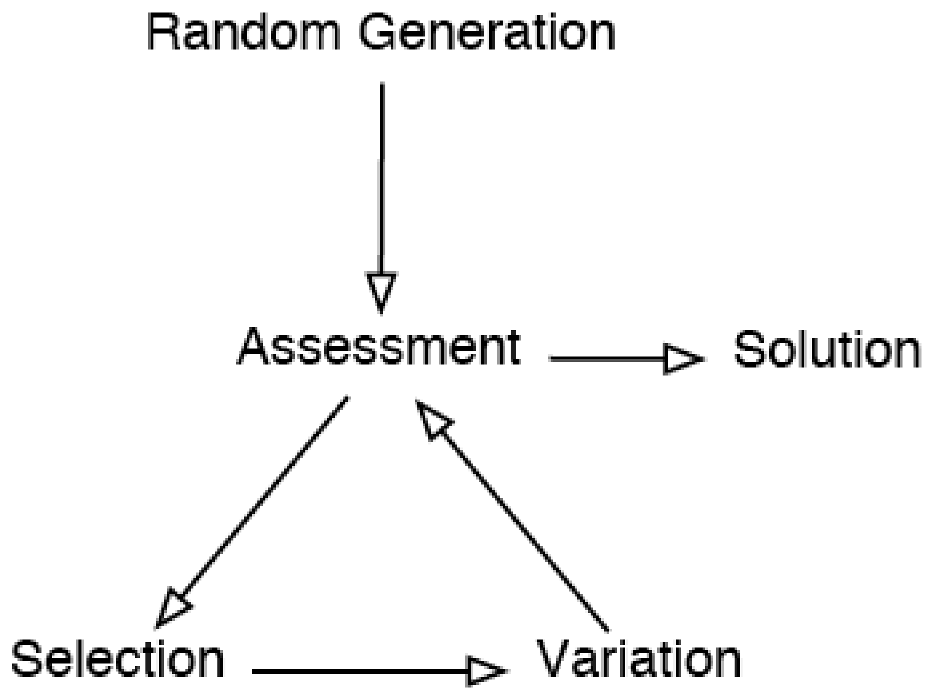
Human Finite Automaton

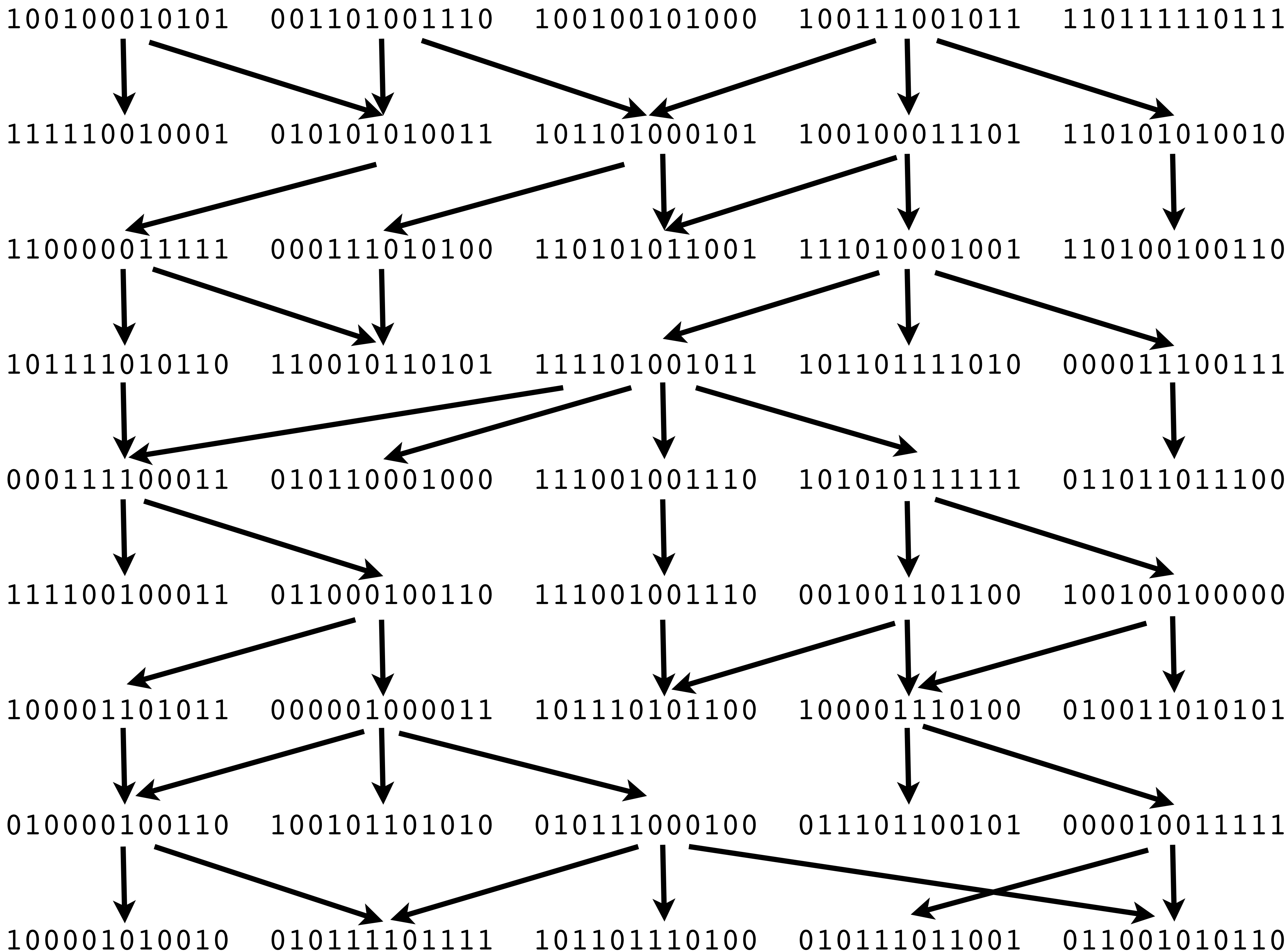
- Random word & target
- Red & blue words & targets
- Choose which of 2 words based on what you've heard

Human Finite Automaton

- Random word & target
- Red & blue words & targets
- Choose which of 2 words based on what you've heard
- Choose any word based on what you've heard

Evolutionary Computation




100100010101
001101001110
100100101000
100111001011
110111110111
111110010001
010101010011
101101000101
100100011101
110101010010
110000011111
000111010100
110101011001
111010001001
110100100110
101111010110
110010110101
111101001011
101101111010
000011100111
000111100011
010110001000
111001001110
101010111111
011011011100
111100100011
011000100110
111001001110
001001101100
100100100000
100001101011
000001000011
101110101100
100001110100
010011010101
010000100110
100101101010
010111000100
011101100101
000010011111
100001010010
010111101111
101101110100
010111011001
011001010110

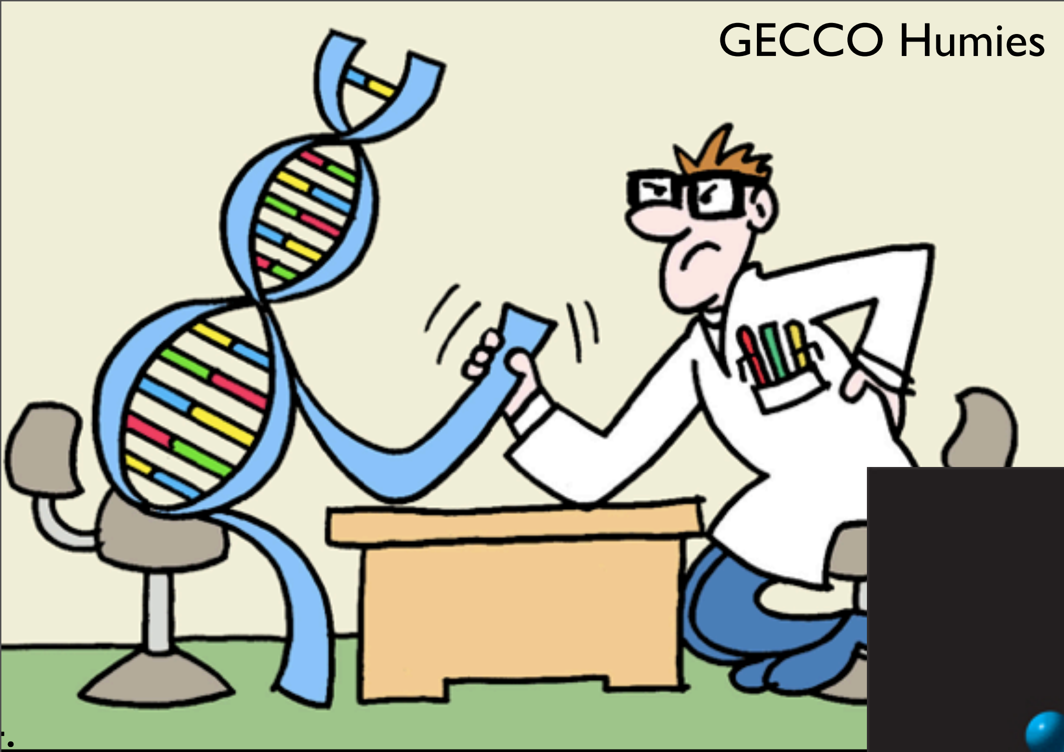
Genetic Programming

- Evolutionary computing to produce executable computer programs.
- Programs are tested by executing them.

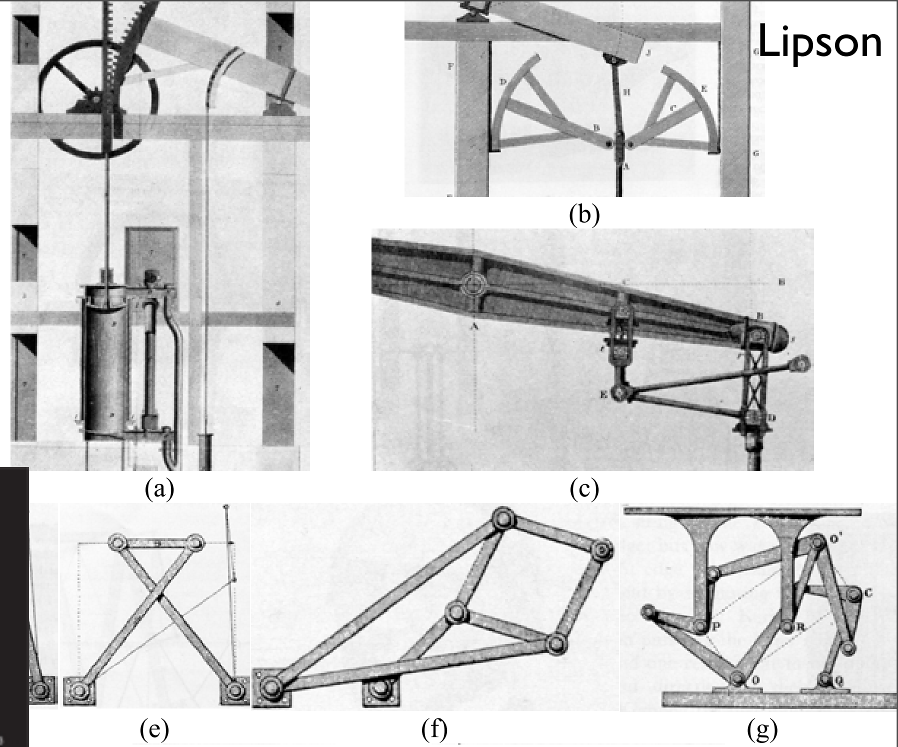
“Gene”tic Programming

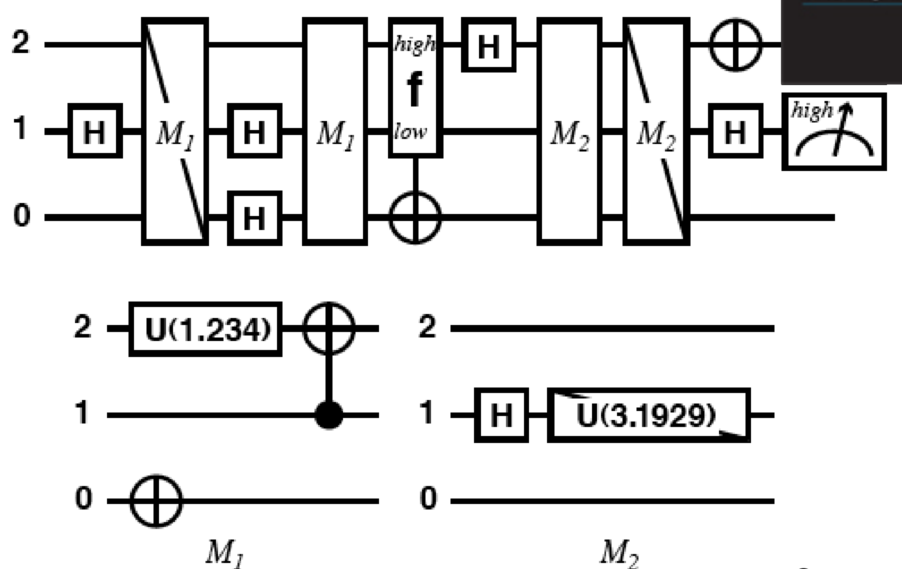
- Mapping between program elements (“genes”) and behavior can be complex
- Some code elements may be “introns”
- Some code elements may act conditionally
- Some code elements may regulate the action of other code elements

GECCO Humies

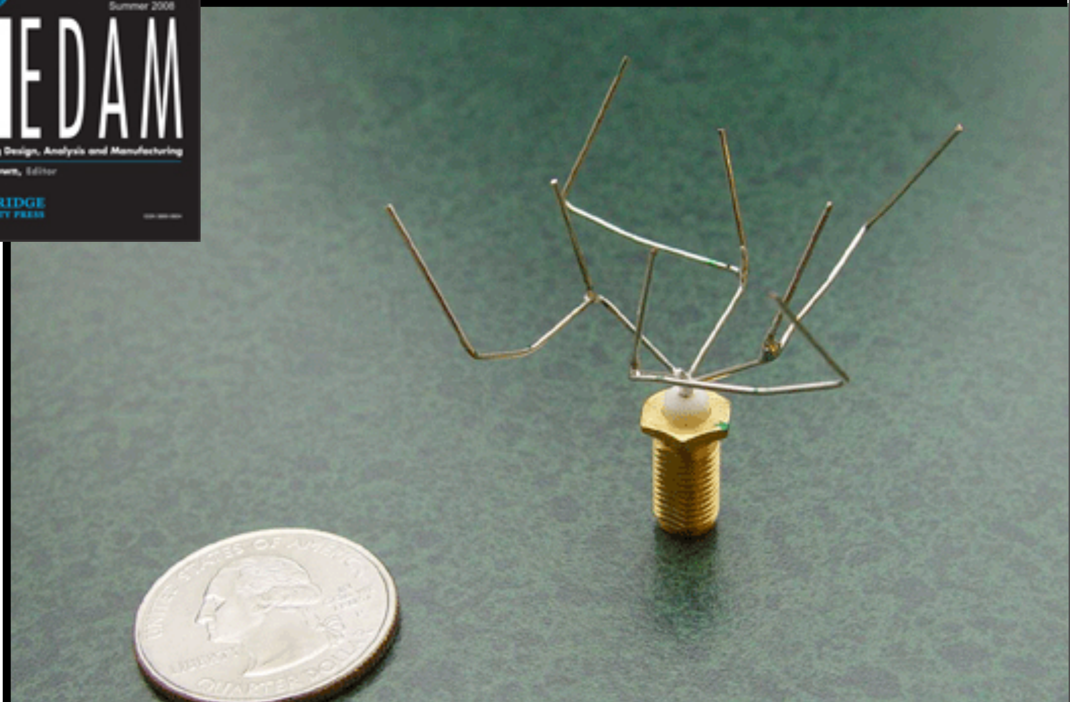


Lipson





Spector



Lohn, Hornby and Linden

Program Representations

- Lisp-style symbolic expressions (Koza, ...).
- Purely functional/lambda expressions (Walsh, Yu, ...).
- Linear sequences of machine/byte code (Nordin et al., ...).
- Artificial assembly-like languages (Ray, Adami, ...).
- Stack-based languages (Perkis, Spector, Stoffel, Tchernev, ...).
- Graph-structured programs (Teller, Globus, ...).
- Object hierarchies (Bruce, Abbott, Schmutter, Lucas, ...)
- Fuzzy rule systems (Tunstel, Jamshidi, ...)
- Logic programs (Osborn, Charif, Lamas, Dubossarsky, ...).
- Strings, grammar-mapped to arbitrary languages (O'Neill, Ryan, ...).

Mutating Lisp

```
(+ (* X Y)
   (+ 4 (- Z 23)))
```

(+ ***(* X Y)***
   (+ 4 (- Z 23)))

```
(+ (- (+ 2 2) Z)
   (+ 4 (- Z 23)))
```

Recombining Lisp

Parent 1:
```
(+ (* X Y)
   (+ 4 (- Z 23)))
```

Parent 2:
```
(- (* 17 (+ 2 X))
   (* (- (* 2 Z) 1)
      (+ 14 (/ Y X))))
```

Child 1:
```
(+ (- (* 2 Z) 1)
   (+ 4 (- Z 23)))
```

Child 2:
```
(- (* 17 (+ 2 X))
   (* (* X Y)
      (+ 14 (/ Y X))))
```

Symbolic Regression

Given a set of data points, evolve a program that produces *y* from *x*.

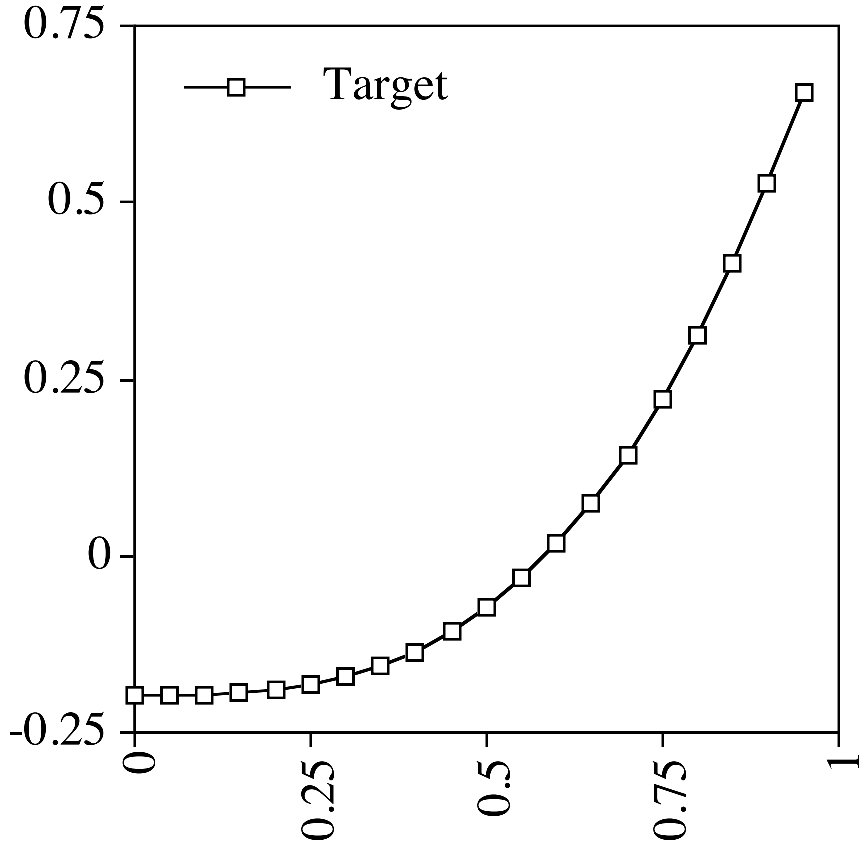
Primordial ooze: +, -, *, %, *x*, 0.1

Fitness = error (smaller is better)

GP Parameters

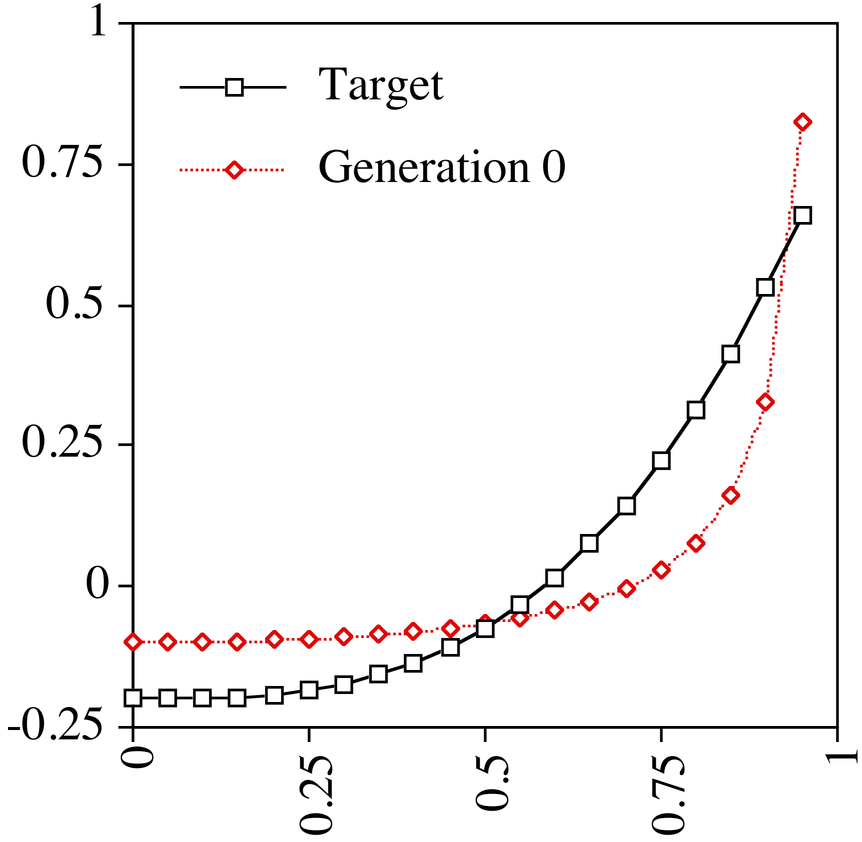
Maximum number of Generations: 51
Size of Population: 1000
Maximum depth of new individuals: 6
Maximum depth of new subtrees for mutants: 4
Maximum depth of individuals after crossover: 17
Fitness-proportionate reproduction fraction: 0.1
Crossover at any point fraction: 0.3
Crossover at function points fraction: 0.5
Selection method: FITNESS-PROPORTIONATE
Generation method: RAMPED-HALF-AND-HALF
Randomizer seed: 1.2

Evolving $y = x^3 - 0.2$



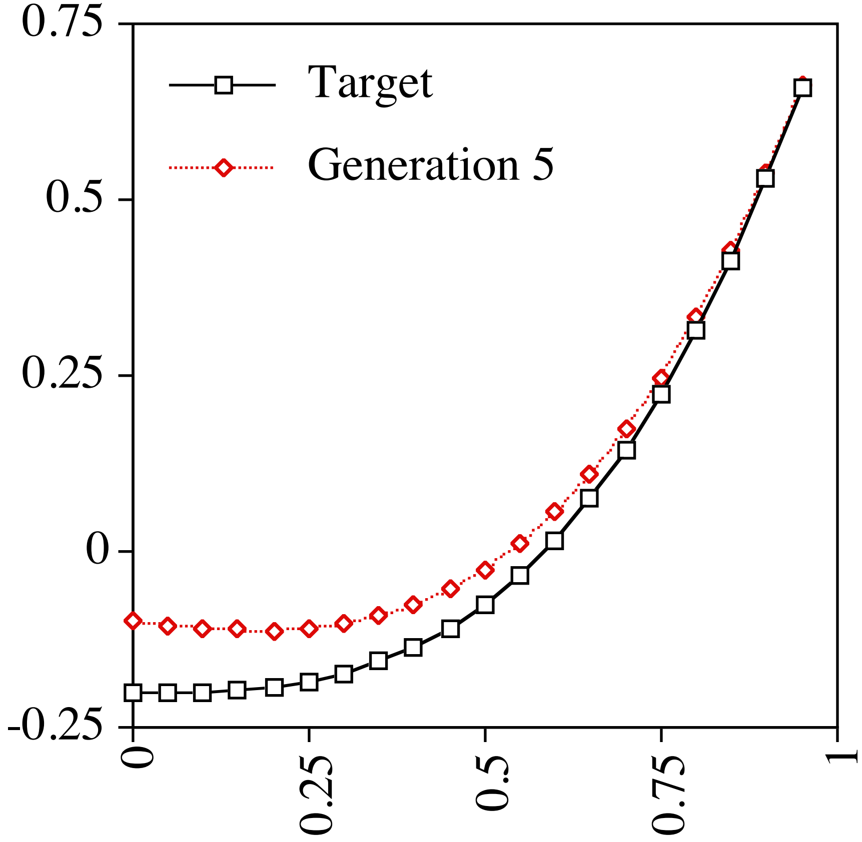
Best Program, Gen 0

```
(- (% (* 0.1
         (* X X))
      (- (% 0.1 0.1)
         (* X X)))
   0.1)
```



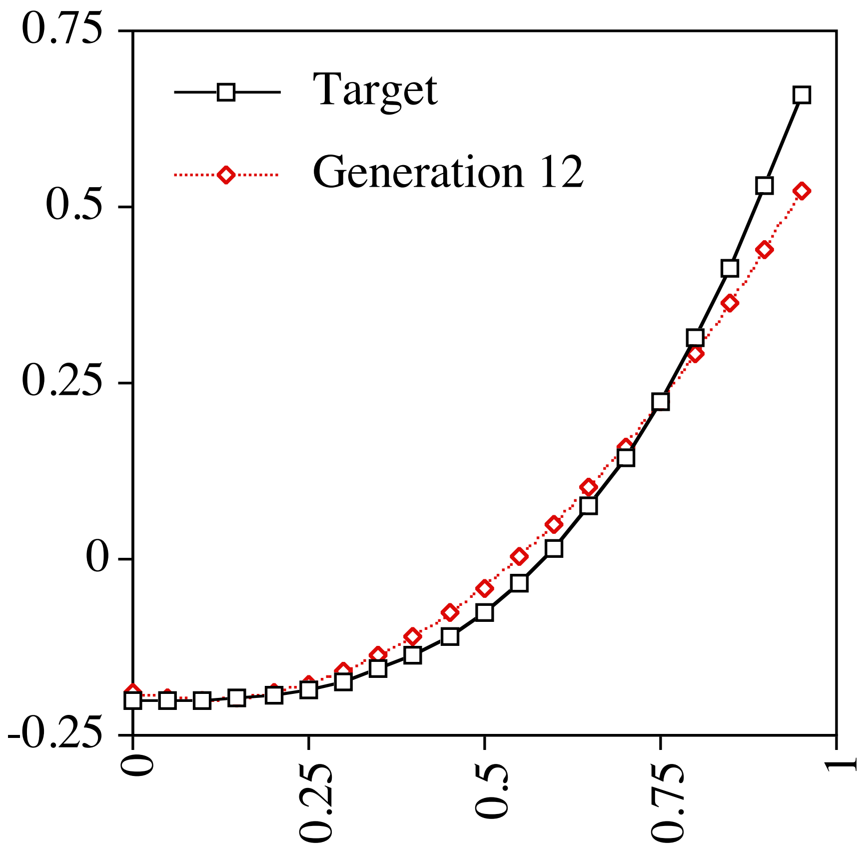
Best Program, Gen 5

```
(- (* (* (% X 0.1)
         (* 0.1 X))
      (- X
         (% 0.1 X)))
   0.1)
```



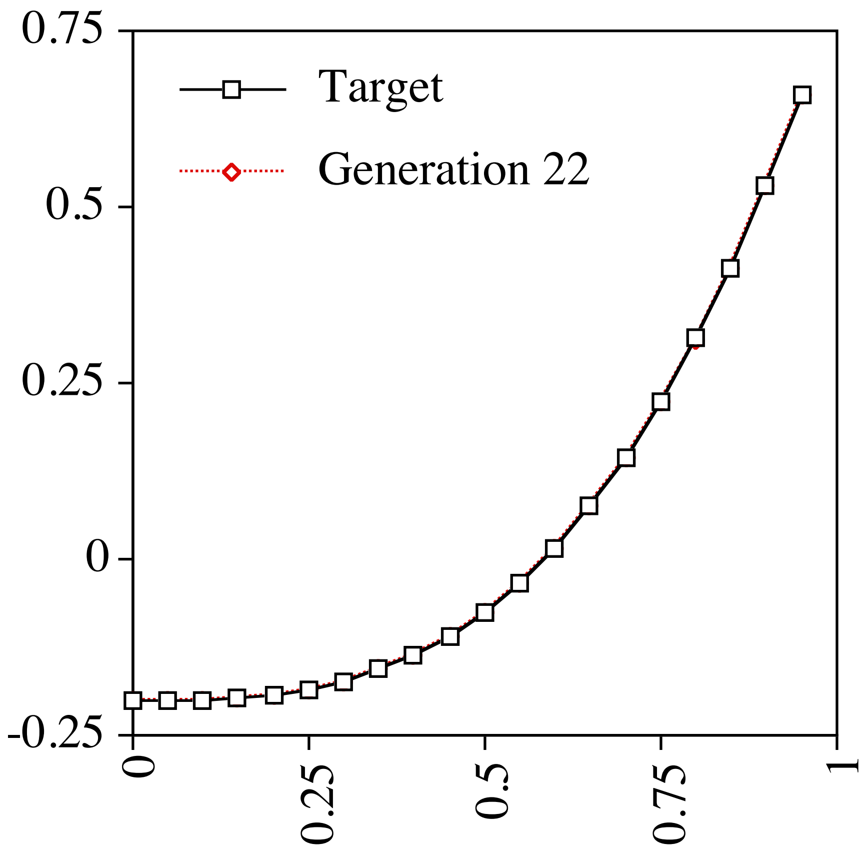
Best Program, Gen 12

```
(+ (- (- 0.1
         (- 0.1
            (- (* X X)
               (+ 0.1
                  (- 0.1
                     (* 0.1
                        0.1))))))
      (* X
         (* (% 0.1
               (% (* (* (- 0.1 0.1)
                        (+ X
                           (- 0.1 0.1)))
                     X)
                  (+ X (+ (- X 0.1)
                          (* X X)))))
            (+ 0.1 (+ 0.1 X)))))
   (* X X))
```



Best Program, Gen 22

```
(- (- (* X (* X X))
      0.1)
   0.1)
```



Genetic Programming for Finite Algebras

Lee Spector
Cognitive Science
Hampshire College
Amherst, MA 01002
lspector@hampshire.edu

David M. Clark
Mathematics
SUNY New Paltz
New Paltz, NY 12561
clarkd@newpaltz.edu

Ian Lindsay
Hampshire College
Amherst, MA 01002
iml04@hampshire.edu

Bradford Barr
Hampshire College
Amherst, MA 01002
bradford.barr@gmail.com

Jon Klein
Hampshire College
Amherst, MA 01002
jk@artificial.com

Humies 2008
GOLD MEDAL

Goal

- Find finite algebra terms that have certain special properties
- For decades there was no way to produce these terms in general, short of exhaustive search
- Current best methods produce enormous terms

Significance, Time

	Uninformed Search Expected Time (Trials)
3 element algebras Mal'cev Pixley/majority discriminator	 5 seconds ($3^{15} \approx 10^7$) 1 hour ($3^{21} \approx 10^{10}$) 1 month ($3^{27} \approx 10^{13}$)
4 element algebras Mal'cev Pixley/majority discriminator	 10^3 years ($4^{28} \approx 10^{17}$) 10^{10} years ($4^{40} \approx 10^{24}$) 10^{24} years ($4^{64} \approx 10^{38}$)

Significance, Time

	Uninformed Search Expected Time (Trials)	GP Time
3 element algebras		
Mal'cev	5 seconds ($3^{15} \approx 10^7$)	1 minute
Pixley/majority	1 hour ($3^{21} \approx 10^{10}$)	3 minutes
discriminator	1 month ($3^{27} \approx 10^{13}$)	5 minutes
4 element algebras		
Mal'cev	10^3 years ($4^{28} \approx 10^{17}$)	30 minutes
Pixley/majority	10^{10} years ($4^{40} \approx 10^{24}$)	2 hours
discriminator	10^{24} years ($4^{64} \approx 10^{38}$)	?

Significance, Size

Term Type	Primality Theorem
Mal'cev	10, 060, 219
Majority	6, 847, 499
Pixley	1, 257, 556, 499
Discriminator	12, 575, 109

(for A_I)

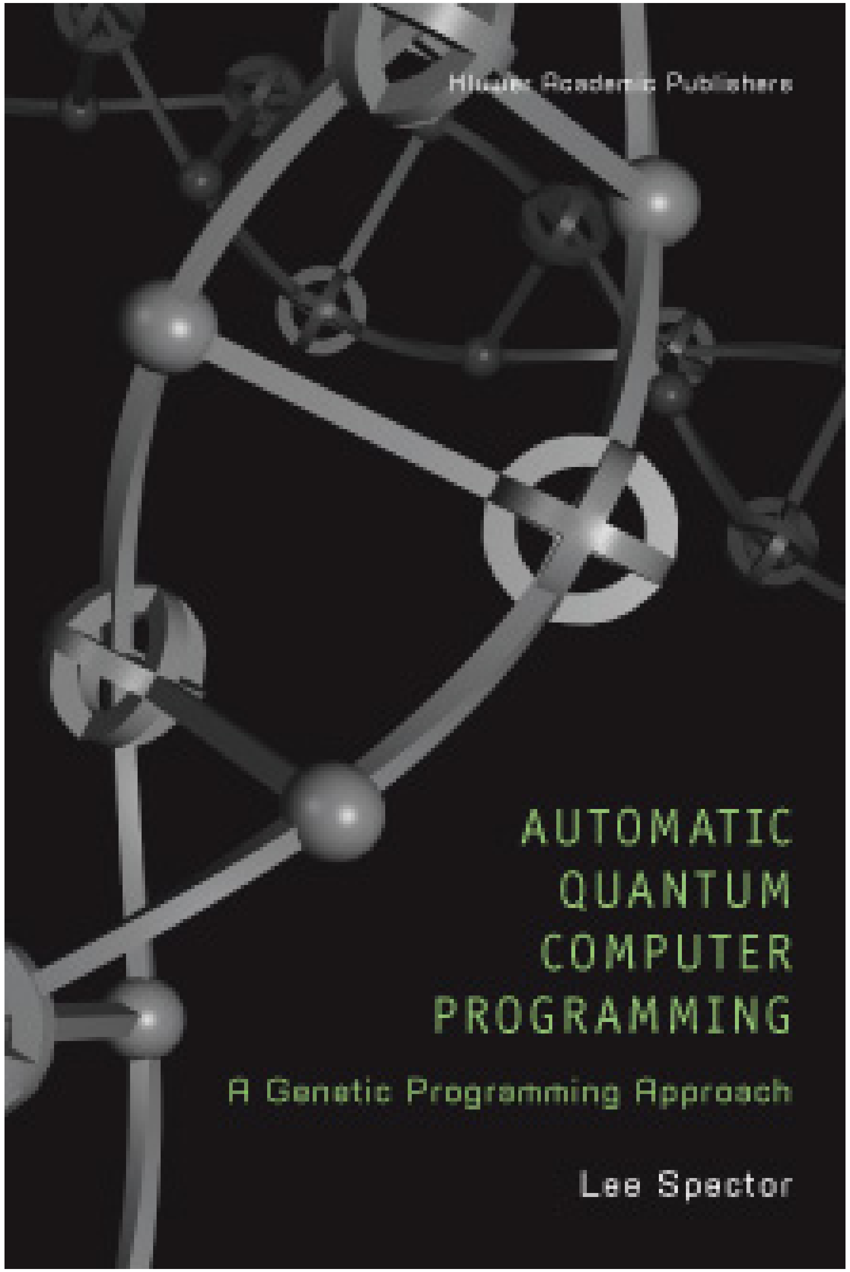
Significance, Size

Term Type	Primality Theorem	GP
Mal'cev	10,060,219	12
Majority	6,847,499	49
Pixley	1,257,556,499	59
Discriminator	12,575,109	39

(for A_I)

Human Competitive?

- Rather: human-**WHOMPING!**
- *Outperforms* humans *and all other known methods* on significant problems, providing benefits of *several orders of magnitude* with respect to search speed and result size
- Because there were no prior methods for generating practical terms in practical amounts of time, GP has provided the first solution to a previously open problem in the field



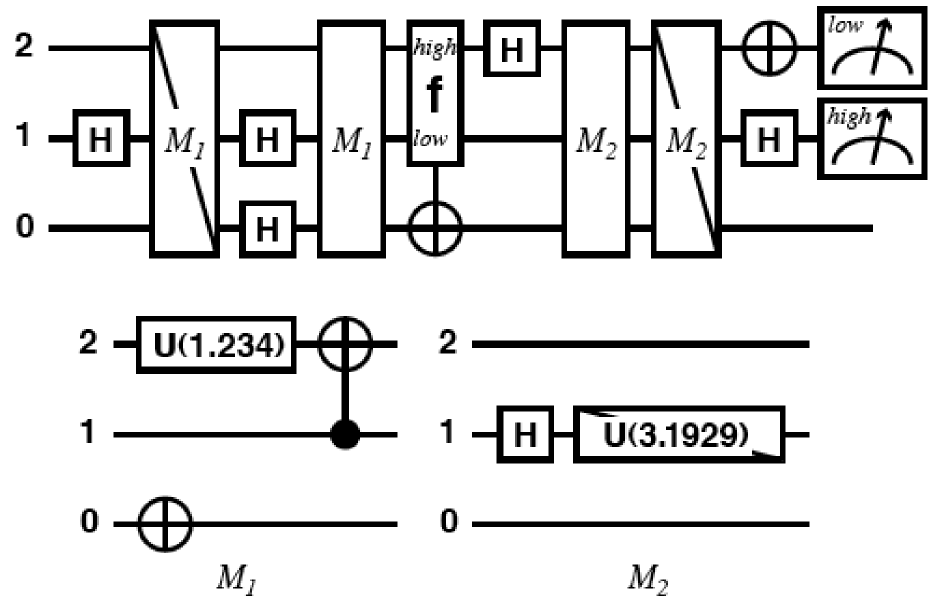


Figure 8.7. A gate array diagram for an evolved version of Grover's database search algorithm for a 4-item database. The full gate array is shown at the top, with M_1 and M_2 standing for the smaller gate arrays shown at the bottom. A diagonal line through a gate symbol indicates that the matrix for the gate is transposed. The "f" gate is the oracle.

Humies 2004 GOLD MEDAL

Evolution, the Designer

"Darwinian evolution is itself a designer worthy of significant respect, if not religious devotion." *Boston Globe* OpEd, Aug 29, 2005

WHAT WOULD DARWIN SAY? | LEE SPECTOR

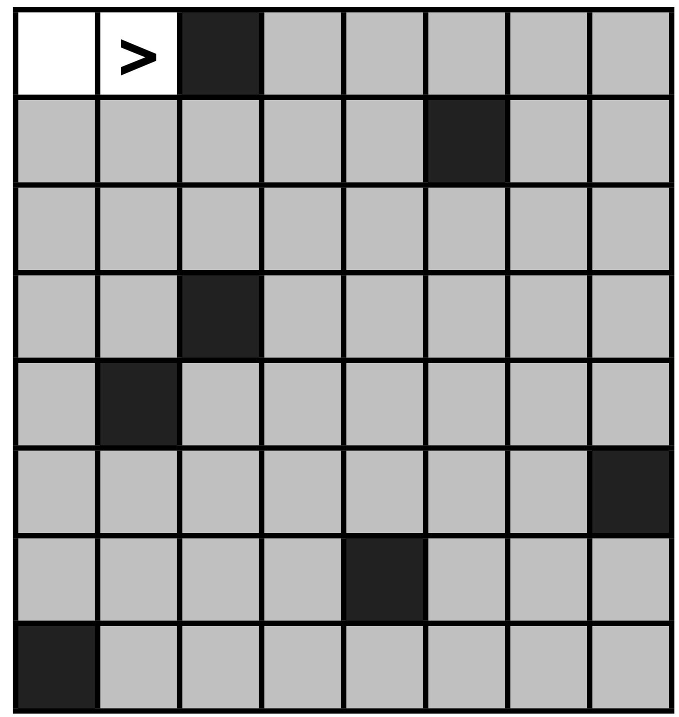
The Boston Globe

And now, digital evolution

By Lee Spector | August 29, 2005

RECENT developments in computer science provide new perspective on "intelligent design," the view that life's complexity could only have arisen through the hand of an intelligent designer. These developments show that complex and useful designs can indeed emerge from random Darwinian processes.

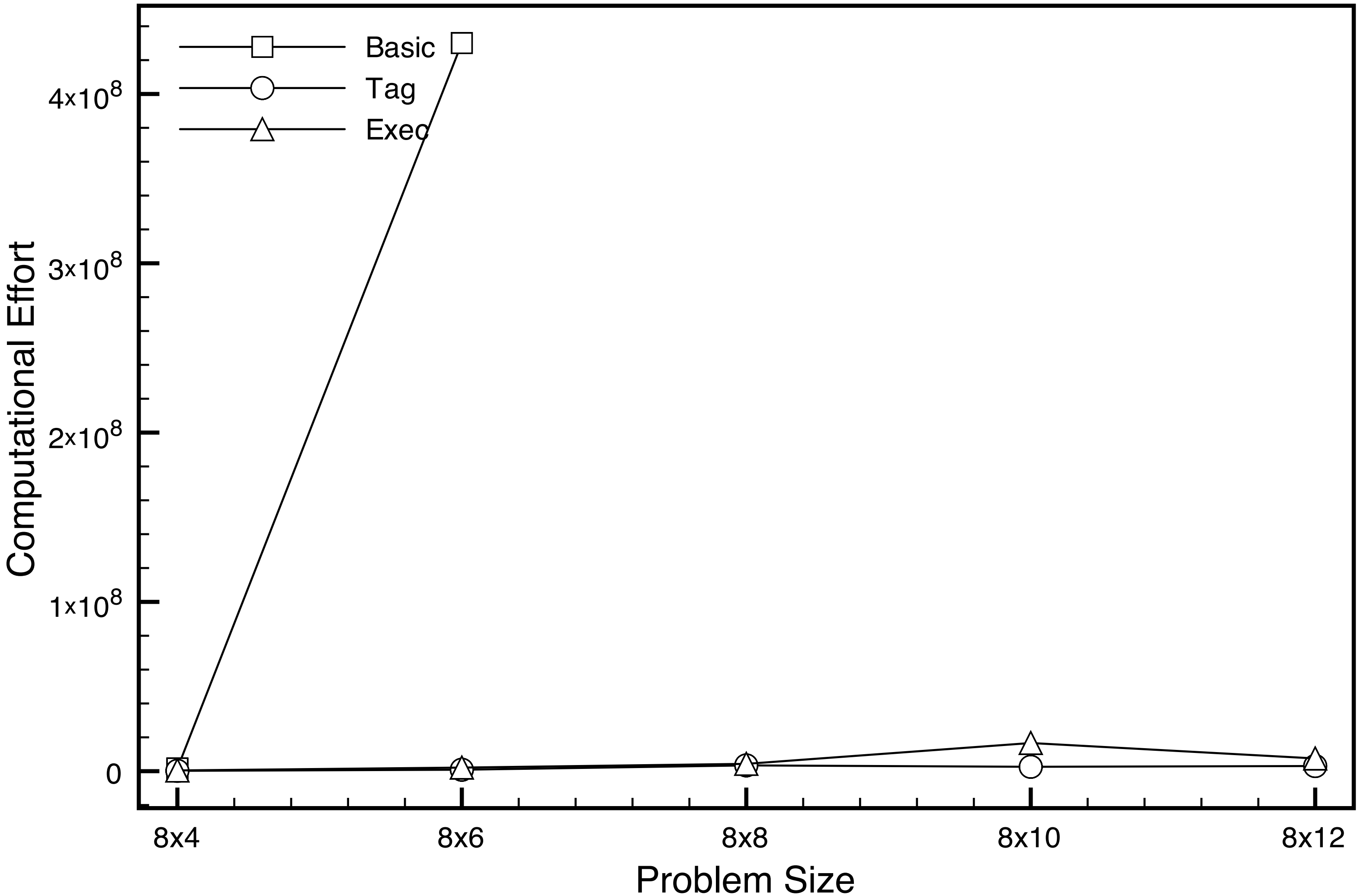
Dirt-Sensing, Obstacle-Avoiding Robot Problem



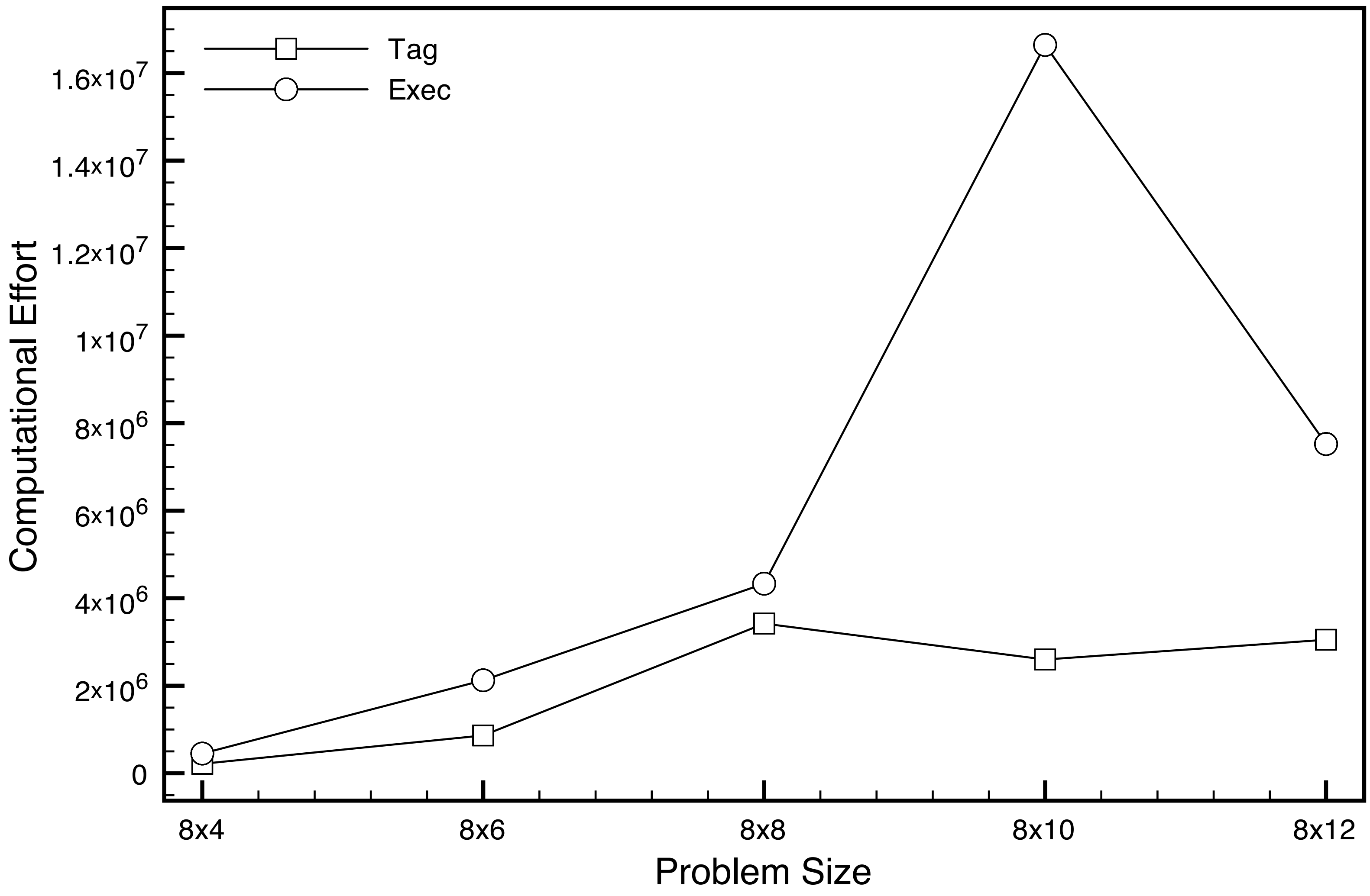
DSOAR Instructions

Condition	Instructions
Basic	if-dirty, if-obstacle, left, mop, v8a, frog, $\mathcal{R}_{v8}$
Tag	if-dirty, if-obstacle, left, mop, v8a, frog, $\mathcal{R}_{v8}$, tag.exec.[1000], tagged.[1000]
Exec	if-dirty, if-obstacle, left, mop, v8a, frog, $\mathcal{R}_{v8}$, exec.dup, exec.pop, exec.rot, exec.swap, exec.k, exec.s, exec.y

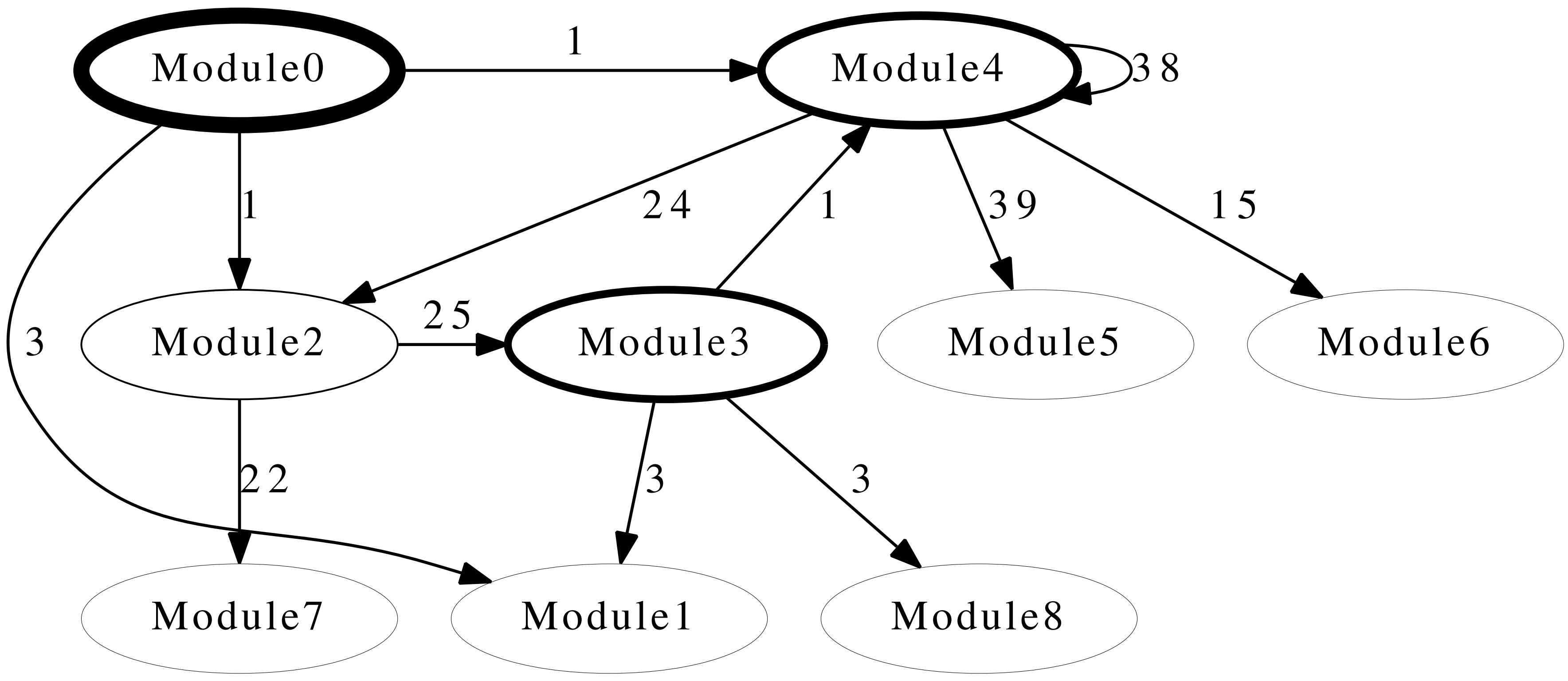
DSOAR Effort



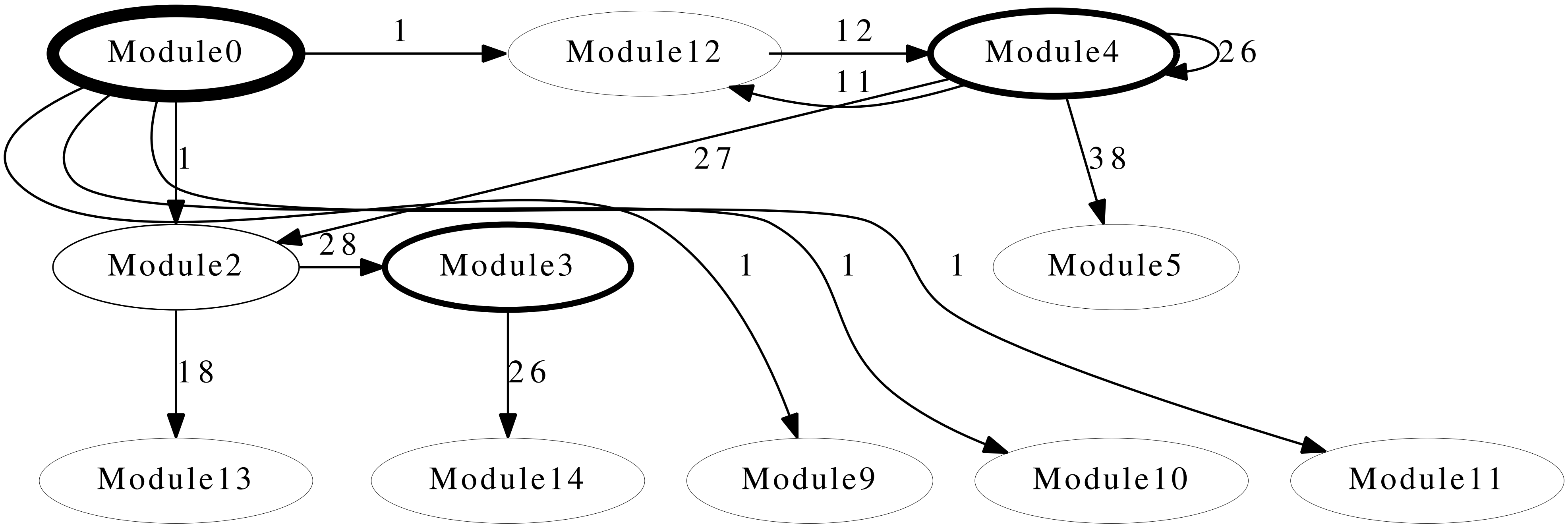
DSOAR Effort



Evolved DSOAR Architecture (in one environment)



Evolved DSOAR Architecture (in another environment)



Autoconstructive Evolution

- Individual programs make their own children
- Hence they control their genetic representations, mutation rates, sexuality, reproductive timing, etc.
- The machinery of reproduction and diversification (i.e., the machinery of evolution) evolves
- Selection may favor reactive and developmental stability

SwarmEvolve 2

Conclusions

- Genetic programming is a powerful problem-solving technique based loosely on biological evolution
- In genetic programming the genome is a reactive system with many features of biological genetic systems that are only now becoming well appreciated, including self regulation and complex interactions between the environment and elements of the genome