

EVOLUTIONARY ALGORITHMS

[Read Chapter 9]

[Exercises 9.1, 9.2, 9.3, 9.4]

- Evolutionary computation
- Prototypical GA
- An example: GABIL
- Schema theorem
- Genetic Programming
- Individual learning and population evolution

Evolutionary Computation

- Biological Evolution

1. Lamarck and others:

Species “transmute” over time

2. Darwin and Wallace:

Consistent, heritable variation among individuals in population

Natural selection of the fittest

3. Mendel and genetics:

A mechanism for inheriting traits

genotype → phenotype mapping

- Computational procedures patterned after biological evolution

- Search procedure that probabilistically applies search operators to set of points in the search space

Simple Genetic Algorithm

$SGA(Fitness, Fitness_threshold, p, r, m)$

- *Initialize*: $P \leftarrow p$ random hypotheses
- *Evaluate*: for each h in P , compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness_threshold$
 1. *Select*: Probabilistically select $(1 - r)p$ members of P to add to P_s .

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^p Fitness(h_j)}$$

2. *Crossover*: Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P . For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s .
 3. *Mutate*: Invert a randomly selected bit in $m \cdot p$ random members of P_s
 4. *Update*: $P \leftarrow P_s$
 5. *Evaluate*: for each h in P , compute $Fitness(h)$
- Return the hypothesis from P that has the highest fitness.

Representing Hypotheses

- Represent

$(Outlook = Overcast \vee Rain) \wedge (Wind = Strong)$

by

<i>Outlook</i>	<i>Wind</i>
011	10

- Represent

IF $Wind = Strong$ THEN $PlayTennis = yes$

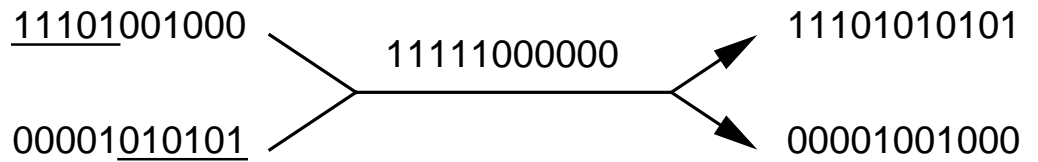
by

<i>Outlook</i>	<i>Wind</i>	<i>PlayTennis</i>
111	10	10

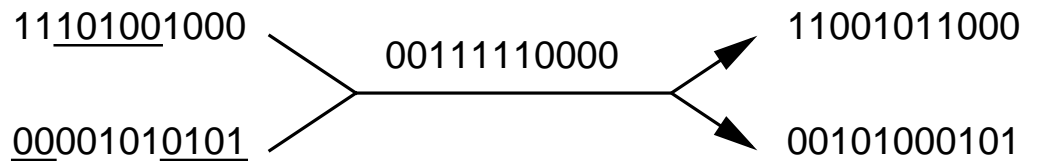
Operators for Genetic Algorithms

Initial strings *Crossover Mask* *Offspring*

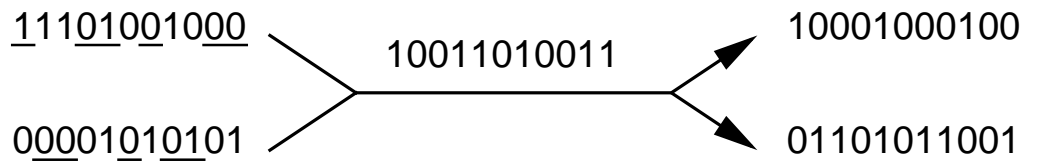
Single-point crossover:



Two-point crossover:



Uniform crossover:



Point mutation:



Selecting Most Fit Hypotheses

Fitness Proportionate Selection:

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^p Fitness(h_j)}$$

... Can lead to *crowding*

Tournament Selection:

- Pick h_1, h_2 at random with uniform prob.
- With probability p , select the more fit.

Rank Selection:

- Sort all hypotheses by fitness
- Prob of selection is proportional to rank

GABIL [DeJong et al. 1993]

- Learn disjunctive set of propositional rules, competitive with C4.5

fitness:

$$Fitness(h) = (correct(h))^2$$

Representation:

IF $a_1 = T \wedge a_2 = F$ THEN $c = T$

IF $a_2 = T$ THEN $c = F$

are represented by

a_1	a_2	c	a_1	a_2	c
10	01	1	11	10	0

Genetic operators: ??

1. Want variable length rule sets
2. Want only well-formed bitstring hypotheses

Crossover with Variable-Length Bitstrings

- Start with

	a_1	a_2	c	a_1	a_2	c
h_1 :	10	01	1	11	10	0

h_2 :	01	11	0	10	01	0
---------	----	----	---	----	----	---

1. Choose crossover points for h_1 , e.g., after bits 1, 8
2. Now restrict points in h_2 to those that produce bitstrings with well-defined semantics, e.g., $\langle 1, 3 \rangle$, $\langle 1, 8 \rangle$, $\langle 6, 8 \rangle$.

- If we choose $\langle 1, 3 \rangle$, result is

	a_1	a_2	c
h_3 :	11	10	0

	a_1	a_2	c	a_1	a_2	c	a_1	a_2	c
h_4 :	00	01	1	11	11	0	10	01	0

GABIL Extensions

- Add new genetic operators, also applied probabilistically:
 1. *AddAlternative*: Generalize constraint on a_i by changing a 0 to 1
 2. *DropCondition*: Generalize constraint on a_i by changing every 0 to 1
- And, add new field to bitstring to determine whether to allow these

a_1	a_2	c	a_1	a_2	c	AA	DC
01	11	0	10	01	0	1	0

- So now the learning strategy also evolves!

GABIL Results

- Performance of GABIL comparable to symbolic rule or tree learning methods C4.5, ID5R, AQ14
- Average performance on a set of 12 synthetic problems:
 1. GABIL without *AA* and *DC* operators: 92.1% accuracy
 2. GABIL with *AA* and *DC* operators: 95.2% accuracy
 3. symbolic learning methods ranged from 91.2 to 96.6

Schemas

- How to characterize evolution of population in GA?
- Schema = string containing 0, 1, * (“don’t care”)
 1. Typical schema: 10**0*
 2. Instances of above schema: 101101, 100000, ...
- Characterize population by number of instances representing each possible schema

$m(s, t)$ = number of instances of schema s in pop
at time t

Consider Just Selection

Let

- $\bar{f}(t)$ = average fitness of population at time t
- $m(s, t)$ = # of instances of schema s at time t
- $\hat{u}(s, t)$ = mean fitness of instances of s at time t

Then

1. Probability of selecting h in one selection step

$$\Pr(h) = \frac{f(h)}{\sum_{i=1}^n f(h_i)} = \frac{f(h)}{n\bar{f}(t)}$$

2. Probability of selecting an instance of s in one step

$$\Pr(h \in s) = \sum_{h \in s \cap p_t} \frac{f(h)}{n\bar{f}(t)} = \frac{\hat{u}(s, t)}{n\bar{f}(t)} m(s, t)$$

3. Expected number of instances of s after n selections

$$E[m(s, t + 1)] = \frac{\hat{u}(s, t)}{\bar{f}(t)} m(s, t)$$

Schema Theorem

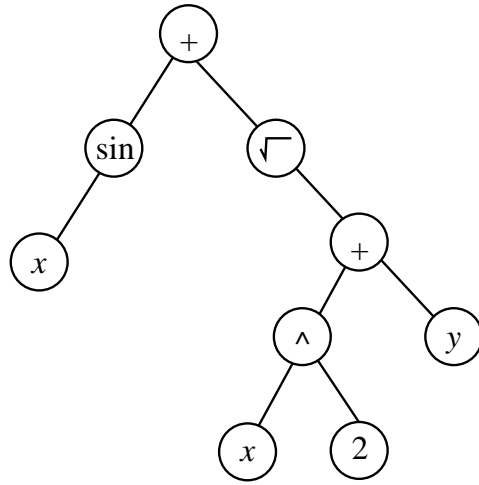
$$E[m(s, t + 1)] \geq \frac{\hat{u}(s, t)}{\bar{f}(t)} m(s, t) \left(1 - p_c \frac{d(s)}{l - 1}\right) (1 - p_m)^{o(s)}$$

- $m(s, t)$ = instances of schema s in pop at time t
- $\bar{f}(t)$ = average fitness of pop. at time t
- $\hat{u}(s, t)$ = ave. fitness of instances of s at time t
- p_c = probability of single point crossover operator
- p_m = probability of mutation operator
- l = length of single bit strings
- $o(s)$ number of defined (non “*”) bits in s
- $d(s)$ = distance between leftmost, rightmost defined bits in s

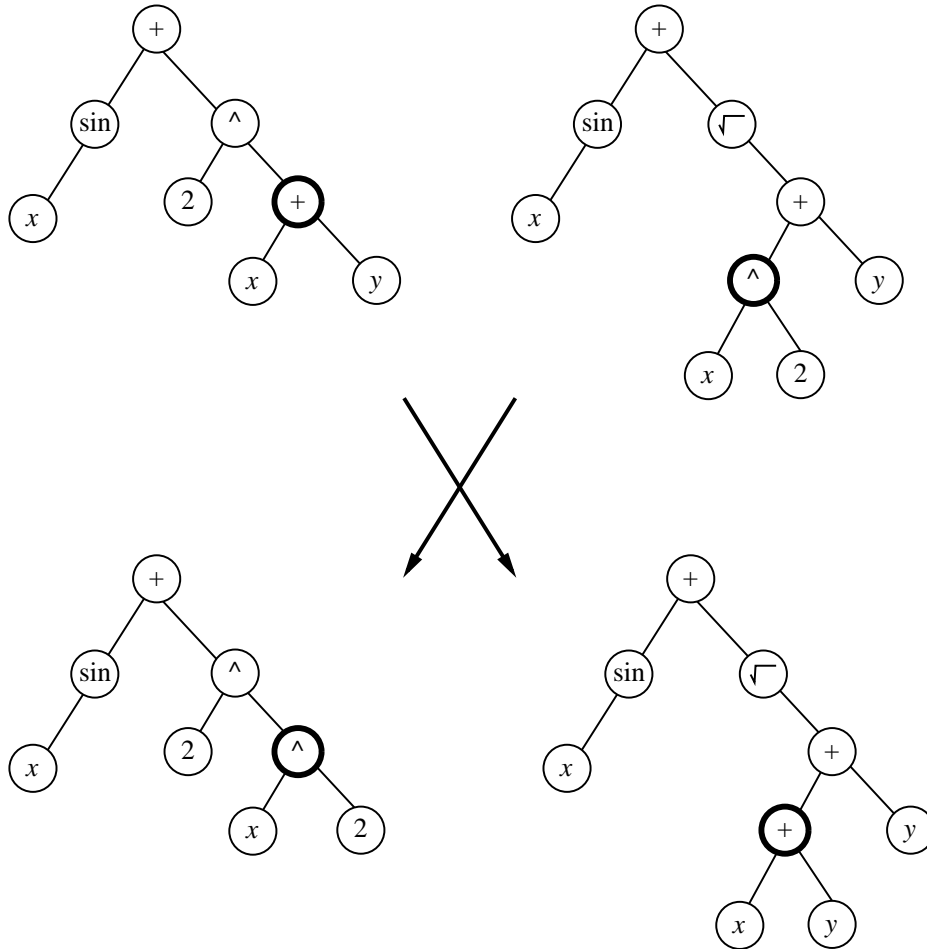
Genetic Programming

Population of programs represented by trees

$$\sin(x) + \sqrt{x^2 + y}$$

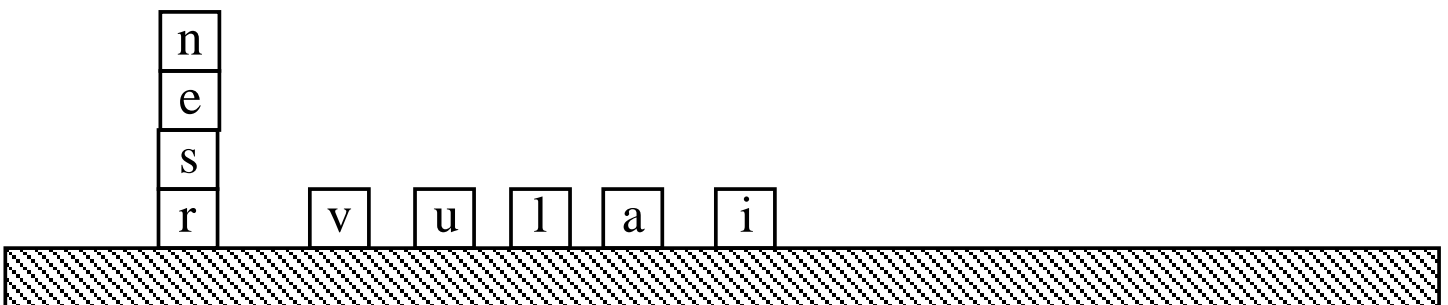


Crossover



Example: GP for Block Word Problem

- Goal: spell UNIVERSAL
- Terminals:
 1. CS (“Current Stack”) = name of the top block on stack, or F .
 2. TB (“Top correct Block”) = name of topmost correct block on stack
 3. NN (“Next Necessary”) = name of the next block needed above TB in the stack



Primitive functions:

- (MS x): (“Move to Stack”), if block x is on the table, moves x to the top of the stack and returns the value T . Otherwise, does nothing and returns the value F .
- (MT x): (“Move to Table”), if block x is somewhere in the stack, moves the block at the top of the stack to the table and returns the value T . Otherwise, returns F .
- (EQ $x y$): (“Equal”), returns T if x equals y , and returns F otherwise.
- (NOT x): Returns T if $x = F$, else returns F
- (DU $x y$): (“Do until”) executes the expression x repeatedly until expression y returns the value T

Learned Program

- Trained to fit 166 test problems
- Using population of 300 programs, found this after 10 generations:

```
(EQ (DU (MT CS)(NOT CS)) (DU (MS NN)(NOT NN)) )
```

- More interesting example

Design electronic filter circuits

1. Individuals are programs that transform beginning circuit to final circuit, by adding/subtracting components and connections
2. Use population of 640,000, run on 64 node parallel processor
3. Discovers circuits competitive with best human designs

GP for Classifying Images

Fitness: Based on coverage and accuracy

Representation:

- Primitives include Add, Sub, Mult, Div, Not, Max, Min, Read, Write, If-Then-Else, Either, Pixel, Least, Most, Ave, Variance, Difference, Mini, Library
- Mini refers to a local subroutine that is separately co-evolved
- Library refers to a global library subroutine (evolved by selecting the most useful minis)

Genetic operators:

- Crossover, mutation
- Create “mating pools” and use rank proportionate reproduction

[Teller and Veloso, 1997]

Biological Evolution

Lamarck (19th century)

- Individual genetic makeup was altered by lifetime experience
- But current evidence contradicts this view

What is the impact of individual learning on population evolution?

Baldwin Effect: Assume

- Individual learning has no direct influence on individual DNA
- But ability to learn reduces need to “hard wire” traits in DNA

Then

1. Ability of individuals to learn will support more diverse gene pool
 - Because learning allows individuals with various “hard wired” traits to be successful
2. More diverse gene pool support faster evolution of gene pool

→ Individual learning (indirectly) increases rate of evolution

Plausible example of Baldwin Effect:

1. New predator appears in environment
2. Individuals who can learn (to avoid it) will be selected
3. Increase in learning individuals will support more diverse gene pool
4. Resulting in faster evolution
5. Possibly resulting in new non-learned traits such as instinctive fear of predator

Computer Experiments on Baldwin Effect

- Evolve simple neural networks:
 1. Some network weights fixed during lifetime, others trainable
 2. Genetic makeup determines which are fixed, and their weight values
- Results:
 1. With no individual learning, population failed to improve over time
 2. When individual learning allowed
 - Early generations:** Population contained many individuals with many trainable weights
 - Later generations:** Higher fitness, while number of trainable weights decreased

[Hinton and Nowlan, 1987]