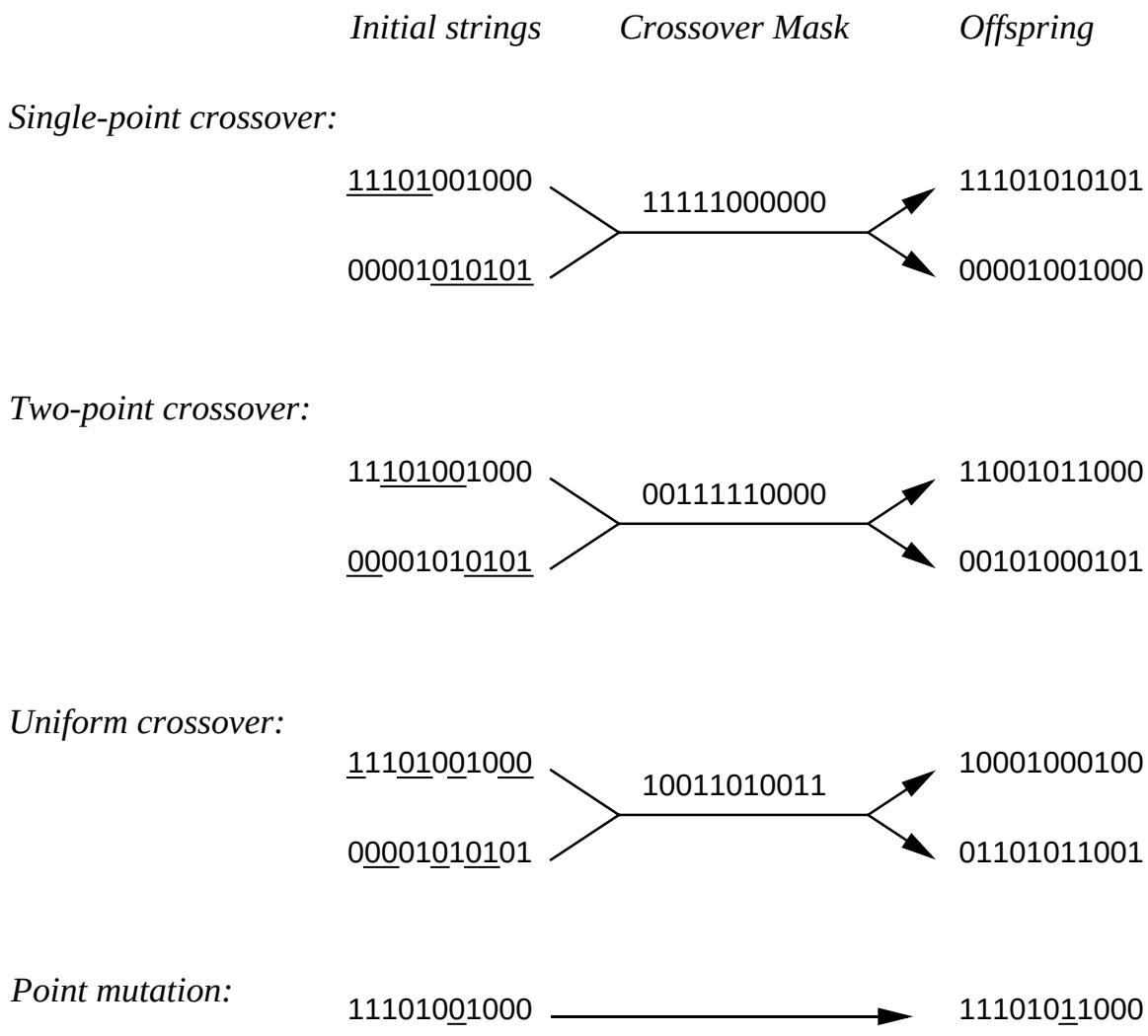
Genetic Algorithms

[Read Chapter 9]
[Exercises 9.1, 9.2, 9.3, 9.4]

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

Operators for Genetic Algorithms



Representing Hypotheses

Represent

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

by

$Outlook$	$Wind$
011	10

Represent

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

by

$Outlook$	$Wind$	$PlayTennis$
111	10	10

遗传算法原型

GA(*Fitness, Fitness_threshold, p, r, m*)

Fitness **:**适应度评分函数，为给定假设赋予一个评估分数

Fitness_threshold **:**指定终止判据的阈值

p **:**群体中包含的假设数量

r **:**每一步中通过交叉取代群体成员的比例

m **:**变异率

- 初始化群体：**P** ← 随机产生的 **p** 个假设
- 评估：对于 **P** 中的每一个 **h**，计算 **Fitness(h)**
- 当 $[\max_h Fitness(h)] < Fitness\_threshold$ ， 做：

  产生新的一代 **Ps:**

  1. 选择：用概率方法选择 **P** 的**(1-r)p** 个成员加入 **Ps**。从 **p** 中选择假设 **hi** 的概率 **Pr(hi)**用下面公式计算：

$$\Pr(hi) = \frac{Fitness(hi)}{\sum_{j=1}^{p} Fitness(hj)}$$

  2. 交叉**:** 根据上面给出的 **Pr(hi)**，从 **p** 中按概率选择 **r.p/2** 对假设。对于每对假设**<h1,h2>**，应用交叉算子产生两个后代。把所有的后代加入 **Ps**
  3. 变异：使用均匀的概率从 **Ps** 中选择 **m%**的成员。对于选出的每个成员，在它的表示中随机选择一位取反
  4. 更新：**P** ← **Ps**
  5. 评估：对于 **P** 中的每个 **h** 计算 **Fitness(h)**

- 从 **P** 中返回适应度最高的假设

个体	染色体	适应度	选择概率	累积概率
1	**0001100000**	**8**	**0.086957**	**0.086957**
2	**0101111001**	**5**	**0.054348**	**0.141304**
3	**0000000101**	**2**	**0.021739**	**0.163043**
4	**1001110100**	**10**	**0.108696**	**0.271739**
5	**1010101010**	**7**	**0.076087**	**0.347826**
6	**1110010110**	**12**	**0.130435**	**0.478261**
7	**1001011011**	**5**	**0.054348**	**0.532609**
8	**1100000001**	**19**	**0.206522**	**0.739130**
9	**1001110100**	**10**	**0.108696**	**0.847826**
10	**0001010011**	**14**	**0.152174**	**1.000000**

随机数序列：**0.0702121, 0.545929, 0.784567, 0.44693, 0.507893**
被选个体：**1, 8, 9, 6, 7**

Selecting Most Fit Hypotheses

Fitness proportionate selection:

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

... 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 $c = F$

represented by

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

Genetic operators: ???

- want variable length rule sets
- 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/tree learning methods C4.5, ID5R, AQ14

Average performance on a set of 12 synthetic problems:

- GABIL without AA and DC operators: 92.1% accuracy
- GABIL with AA and DC operators: 95.2% accuracy
- 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")

- Typical schema: 10**0*
- 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

- $\bar{f}(t) =$ average fitness of pop. at time t
- $m(s,t) =$ instances of schema s in pop at time t
- $\hat{u}(s,t) =$ ave. fitness of instances of s at time t

Probability of selecting h in one selection step

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

Probabilty of selecting an instance of s in one step

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

Expected number of instances of s after n selections

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

交叉操作对模式的影响：

A = 0 1 1 1 0 0 0

S1= * 1 * * * * 0

S2= * * * 1 0 * *

A = 0 1 1 | 1 0 0 0

S1= * 1 *| * * * 0

S2= * * *| 1 0 * *

S1 比 **S2** 更不易生存

模式 **S** 被破坏的概率为 **d(S)/(L-1)**， 生存的概率为 **1-d(S)/(L-1)**

被挑选做交叉的概率 **Pc**

模式 **S** 生存的概率计算如下：

$$Ps \geq 1 - Pc \bullet \frac{d(S)}{L-1}$$

变异操作对模式的影响：

以概率 **Pm** 随机改变一个位上的值，单个位存活的概率为**(1-Pm).**

设模式 **S** 有 **O(S)**个确定位，模式 **S** 被保留下来的概率为：

$$(1-Pm)^{O(S)}$$

Schema Theorem

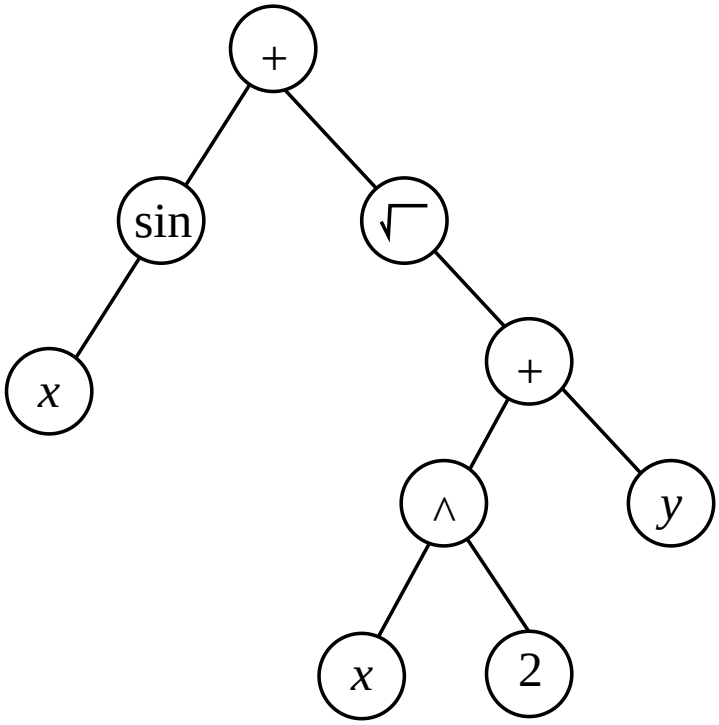
$$E[m(s,t+1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t) \left(1 - p_c \frac{d(s)}{l-1}\right) (1-p_m)^{o(s)}$$

- $m(s,t)$ = 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

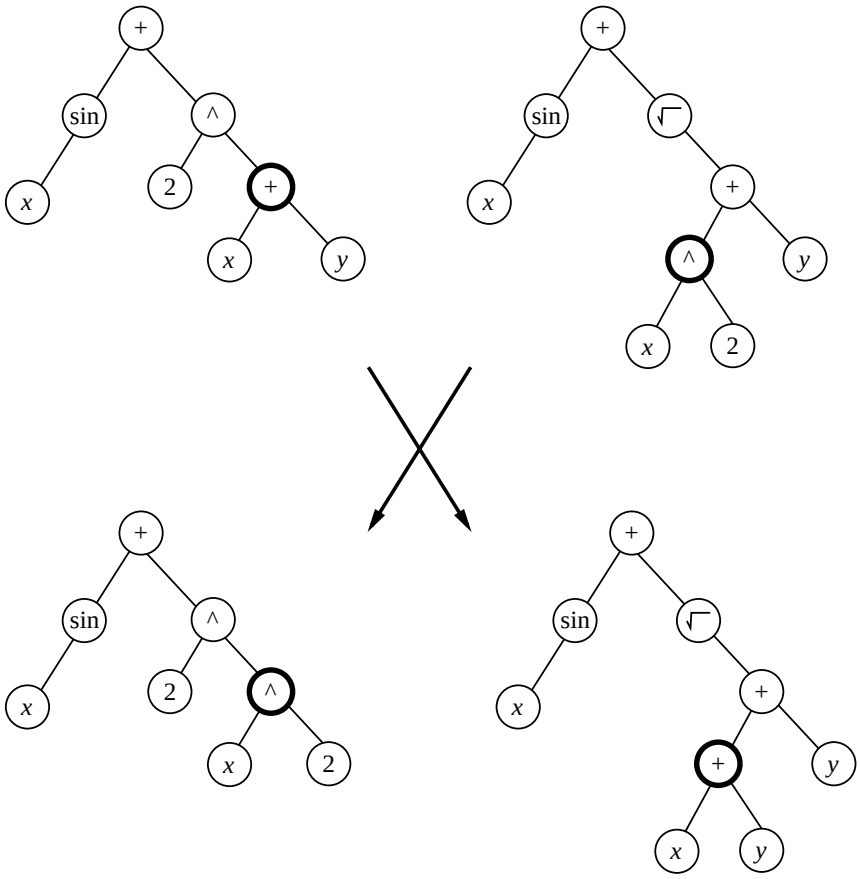
Genetic Programming

Population of programs represented by trees

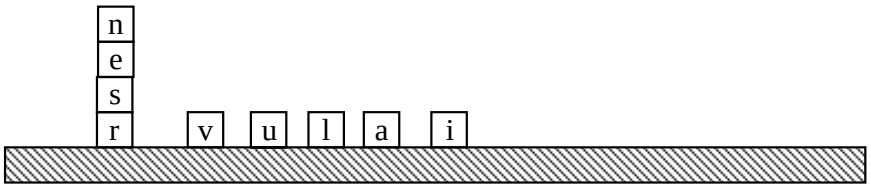
$$\sin(x) + \sqrt{x^2 + y}$$



Crossover



Block Problem



Goal: spell UNIVERSAL

Terminals:

- CS ("current stack") = name of the top block on stack, or F.
- TB ("top correct block") = name of topmost correct block on stack
- 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)))

Genetic Programming

More interesting example: design electronic filter circuits

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

GP for Classifying Images

[Teller and Veloso, 1997]

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

Biological Evolution

Lamark (19th century)

- Believed 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

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

$\rightarrow$ individual learning (indirectly) increases rate of evolution

Baldwin Effect

Plausible example:

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 instintive fear of predator

Computer Experiments on Baldwin Effect

[Hinton and Nowlan, 1987]

Evolve simple neural networks:

- Some network weights fixed during lifetime, others trainable
- Genetic makeup determines which are fixed, and their weight values

Results:

- With no individual learning, population failed to improve over time
- When individual learning allowed
  - Early generations: population contained many individuals with many trainable weights
  - Later generations: higher fitness, while number of trainable weights decreased

Summary: Evolutionary Programming

- Conduct randomized, parallel, hill-climbing search through H
- Approach learning as optimization problem (optimize fitness)
- Nice feature: evaluation of Fitness can be very indirect
  - consider learning rule set for multistep decision making
  - no issue of assigning credit/blame to indiv. steps