Genetic Algorithms

• Connecting evolution and learning
  – Apply evolutionary adaptation to computational problem solving
  – Problem solving as search
    • Not traditional A.I. search: heuristics + backtracking
    • Search with a population of agents

• Principles borrowed from evolution
  – Natural selection - survival of the fittest
    • fit individuals produce more progeny
  – Breeding
    • Novelty from the recombination of existing components
    • Cross-over of two individuals’ chromosomes
  – Mutation
    • Novelty by introducing new genetic material
  – Many individuals comprise system
    • Like connectionism - mass behavior
    • Unlike connectionism - individuals are symbolic strings
    • Unlike connectionism - not much interaction among individuals - each individual is a complete solution the problem
Searching for good solutions on a rough landscape
The Power of Building Blocks

Holistic Versus Analytic Representations
Properties of Genetic Algorithms

- Symbolic codes: each individual represented by a string
- Search via biased sampling from population, not blind search
- Fitness function determines reproduction rates
  - GA performs best if fitness function is graded
- Implicit parallelism
  - each string has many schema in it
  - \( N^3 \) schemas searched in a population of \( N \) individuals
- GA works by increasing prevalence of successful schema
  - exploration versus exploitation
    - Explore: find new solutions
    - Exploit: take advantage of previously found solutions
  - Two-armed bandit problem
  - Probability matching
Abstraction borrowed from biology

- Chromosome: entire symbolic string that defines an individual
- Gene: a feature or characteristic (e.g. color, size, or gender)
- Allele: a value of a feature (e.g. red, large, or male)
- Locus: a position along the string
- Epistasis: interaction between feature values (producing a rough fitness landscape and a difficult search)
  - Crossover
  - Mutation
The Essential Genetic Algorithm

- Initial population of chromosomes
  - Each number corresponds to a property of an "organism" representing a possible solution to the task

- Assess fitness - how well each chromosome carries out the task

- Kill poorest performers

- Breed chromosomes
  - Crossover

- Stop when one chromosome achieves a predetermined fitness

- Random mutation

- New population of chromosomes
### A Simple GA example

<table>
<thead>
<tr>
<th>String No.</th>
<th>Initial Population</th>
<th>x Value (Unsigned Integer)</th>
<th>$f(x)$</th>
<th>pselect, $\frac{f}{\sum f}$</th>
<th>Expected count</th>
<th>Actual Count from Roulette Wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 1 1 0 1</td>
<td>13</td>
<td>169</td>
<td>0.14</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>2</td>
<td>1 1 0 0 0</td>
<td>24</td>
<td>576</td>
<td>0.49</td>
<td>1.97</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0 0 0</td>
<td>8</td>
<td>64</td>
<td>0.06</td>
<td>0.22</td>
<td>0.24</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 1 1</td>
<td>19</td>
<td>361</td>
<td>0.31</td>
<td>1.23</td>
<td>1.24</td>
</tr>
</tbody>
</table>

| Sum        | 1170               | 1.00                        | 4.00   | 4.0                           |
| Average    | 293                | 0.25                        | 1.00   | 1.0                           |
| Max        | 576                | 0.49                        | 1.97   | 2.0                           |

<table>
<thead>
<tr>
<th>Mating Pool after Reproduction (Cross Site Shown)</th>
<th>Mate (Randomly Selected)</th>
<th>Crossover Site (Randomly Selected)</th>
<th>New Population</th>
<th>x Value</th>
<th>$f(x)$</th>
<th>$x^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 1</td>
<td>2</td>
<td>4</td>
<td>0 1 1 0 0</td>
<td>12</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 0</td>
<td>1</td>
<td>4</td>
<td>1 1 0 0 1</td>
<td>25</td>
<td>625</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 0</td>
<td>4</td>
<td>2</td>
<td>1 1 0 1 1</td>
<td>27</td>
<td>729</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1 1</td>
<td>3</td>
<td>2</td>
<td>1 0 0 0 0</td>
<td>16</td>
<td>256</td>
<td></td>
</tr>
</tbody>
</table>

Total $1754$

| 439 |
| 729 |
Schema

• Schema = a substring within an individual
  – Each schema is a possible reason why the individual has the fitness that it does

• Every string of length K has $2^k$ schema in it

• The string 111 has the following schemas
  11*
  **1
  111
  1*1
  *1*
  ***
  *11
  1**

  – * = wildcard - can be filled with any symbol
  – 111 is the most specific schema, and *** the most general

• Every individual tests $2^k$ hypotheses

• The total number of schemas possible in a population of strings with K D-valued features is $(D+1)^k$
First Fundamental Theorem of GAs

- \( P(o, t+1) \geq [1 - a(o, t)] [1 - P_{\text{mut}}]^{d(o)} [U(o, t)/U(t)] P(o, t) \)
  - \( P(o, t+1) \) = fraction of the population occupied by instances of schema \( o \) at time \( t+1 \)
  - \( [U(o, t)/U(t)] \) = ratio of average fitness value of individuals with schema \( o \) compared to average fitness in the whole population
  - \( P(o, t) \) because next generation based on current generation
  - \( a(o, t) = P_{\text{cross}} \frac{L(o)}{K}/P(o, t) \)
    - \( P_{\text{cross}} \) = probability that a selected pair of individuals will be crossed
    - \( L(o) = \) length of schema \( o \), \( K = \) total length of string
    - \( P(o, t) \) because if schema is prevalent, then cross-over will usually just swap the schema with a replica of itself, causing no change
    - \( [1 - a(o, t)] \) = probability of preserving schema despite cross-over potential
  - \( P_{\text{mut}} = \) probability of mutation
    - \( d(o) = \) number of critical, defining, loci in schema \( o \)
    - \( [1 - P_{\text{mut}}]^{d(o)} = \) probability of preserving schema despite potential for mutation

- If \( U(o, t) \) is above average, then schema \( o \) will tend to increase in prevalence
Schema Length and Perpetuation

Schema 1:

| 11111111111111111111111111111111 |

| 3 |

Schema has 3 interior crossover points, so there are 3 chances in 50 that a randomly chosen crossover point will fall in the schema's interior.

Schema 2:

| 11111111111111111111111111111111 |

| 20 |

Schema has 20 interior crossover points, so there are 20 chances in 50 that a randomly chosen crossover point will fall in the schema's interior.

Instance of schema 2 destroyed by crossover:

| 1110 | 0 | 11111111111111111111111111111111 |

| point of crossover |

| 11111111111111111111111111111111 |

| 01110 |

If values (alleles) at the defining positions for a schema are the same on both chromosomes, then the schema will not be disrupted, even if the crossover point falls within the outer limits of the schema:

| 11111111111111111111111111111111 |

| 1100 |

| 01111111111111111111111111111111 |

| 1100 |

Valuable addition to GA: Rearrange genes so that alleles that need to work together form a compact schema.
Second Fundamental Theorem of GAs

- Select some bound $e$ on the transcription error under reproduction and crossover, and pick $L$ such that $L/K \leq e/2$, then in a population of size $M$, where

$$M = C2^{K/L}$$

obtained as a uniform random sample from $\{1,0\}^K$, the number of $L$-length schemas propagated with an error less than $e$ exceeds $M^3$

- $L =$ contiguous defining loci of a schema
- $K =$ length of chromosome (total length of string)
- $M =$ population size
- An average of $C$ instances of each schema in population

- As $L$ increases, we need a bigger population to test the schemas
- If population is large enough, then many schemas are simultaneously tested
- $N^3$ schemas tested with only $N$ fitness function evaluations
Traveling Salesperson Problem
Initial Population for TSP

(5,3,4,6,2)  (2,4,6,3,5)  (4,3,6,5,2)
(2,3,4,6,5)  (4,3,6,2,5)  (3,4,5,2,6)
(3,5,4,6,2)  (4,5,3,6,2)  (5,4,2,3,6)
(4,6,3,2,5)  (3,4,2,6,5)  (3,6,5,1,4)
Select Parents

(5,3,4,6,2)  (2,4,6,3,5)  (4,3,6,5,2)
(2,3,4,6,5)  (4,3,6,2,5)  (3,4,5,2,6)
(3,5,4,6,2)  (4,5,3,6,2)  (5,4,2,3,6)
(4,6,3,2,5)  (3,4,2,6,5)  (3,6,5,1,4)

Try to pick the better ones.
Create Off-Spring with crossover

- (5,3,4,6,2)
- (2,3,4,6,5)
- (3,5,4,6,2)
- (4,6,3,2,5)
- (2,4,6,3,5)
- (4,3,6,2,5)
- (3,4,6,2,5)
- (4,5,3,6,2)
- (5,4,2,3,6)
- (3,4,2,6,5)
- (3,6,5,1,4)

(3,4,5,6,2)
Create More Offspring
Mutate

(5,3,4,6,2)  (2,4,6,3,5)  (4,3,6,5,2)
(2,3,4,6,5)  (4,3,6,2,5)  (3,4,5,2,6)
(3,5,4,6,2)  (4,5,3,6,2)  (5,4,2,3,6)
(4,6,3,2,5)  (3,4,2,6,5)  (3,6,5,1,4)
(3,4,5,6,2)  (5,4,2,6,3)
Mutate

Swap
12345->
14325

Inversion
12345->
14325

Move
12345->
13425
Eliminate

Tend to kill off the worst ones.
Integrate

<table>
<thead>
<tr>
<th>(5,3,4,6,2)</th>
<th>(2,4,6,3,5)</th>
<th>(5,4,2,6,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,4,5,6,2)</td>
<td>(2,3,6,4,5)</td>
<td>(3,4,5,2,6)</td>
</tr>
<tr>
<td>(3,5,4,6,2)</td>
<td>(4,5,3,6,2)</td>
<td>(5,4,2,3,6)</td>
</tr>
<tr>
<td>(4,6,3,2,5)</td>
<td>(3,4,2,6,5)</td>
<td>(3,6,5,1,4)</td>
</tr>
</tbody>
</table>
**Restart**

<table>
<thead>
<tr>
<th>(5,3,4,6,2)</th>
<th>(2,4,6,3,5)</th>
<th>(5,4,2,6,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,4,5,6,2)</td>
<td>(2,3,6,4,5)</td>
<td>(3,4,5,2,6)</td>
</tr>
<tr>
<td>(3,5,4,6,2)</td>
<td>(4,5,3,6,2)</td>
<td>(5,4,2,3,6)</td>
</tr>
<tr>
<td>(4,6,3,2,5)</td>
<td>(3,4,2,6,5)</td>
<td>(3,6,5,1,4)</td>
</tr>
</tbody>
</table>
Additional Complexities and Possibilities

- Individuals in a population frequently converge too quickly
  - If most individuals are similar, then search is inefficient because of redundancy
  - Solution 1: resource sharing. Benefit to finding uncommon solution (Holland, 1975)
  - Solution 2: crowding = individuals replace existing individuals according to their similarity (De Jong, 1985)

- Hybrids that blend dissimilar solutions are usually unfit
  - Blending between mutually exclusive strategies
    - Carnivore: large, Sharp teeth, eyes moved to front, digest fats
    - Herbivore: small (hide), blunt teeth, eyes spread out, digest grass
    - Hybrid: small, sharp teeth, eyes spread out, digest fats
  - Solution 1: Increase likelihood of mating if similar (Booker, 1985)
  - Solution 2: Inbreeding until family average fitness ceases to increase, then crossbreeding (Hollstien, 1971)
  - In biology, similars, not opposites, generally attract
Resource Sharing

All individuals who choose a particular strategy must share the payoffs from the strategy with each other.

Ideal Free Distribution (Fretwell & Lucas, 1970): If optimal, the distribution of agents will match the distribution of payoffs.

Decrease fitness as a function of similarity to other strings (Goldberg & Richardson, 1987)
Resource Sharing

Sharing, No Mutation. 
Generation 100.

(a) With sharing

No Sharing, No Mutation. 
Generation 100.

(b) No sharing
Classifier Systems (Holland)

- Classifiers
  - condition/action rules
  - “If slobbers, four-legged, and furry, then output dog classification”
  - Message = action
  - K-bit binary strings typically
  - Same role/slot format used in GAs

- Execution cycle
  1. Add messages from input interface to message list
  2. Compare all messages on message list to conditions of all classifiers, and record matches
  3. For each match satisfying the condition part of a classifier, post the message specified by its action part to a list of new messages
  4. Replace all old messages by new messages
  5. Translate messages to output interface
### Classifiers

**MESSAGE LIST**

- k-bit strings
- `10011011`
- `00100000`
- ...
- `11101100`

**RULE LIST**

- `cond_1, cond_2 / message [strength]`
- `1#####, ####00# / 00110000 [68]`
- `11111111, 00000000 / 11111111 [83]`
- ...
- `#00110# , 00#### / 11111111 [240]`

- specifies subset of messages
- adjusted by credit assignment (bucket brigade) to reflect average "usefulness"
Classifiers
Bucket-brigade Algorithm

- **Credit assignment problem**
  - How do we determine which classifiers were responsible for a given outcome?
  - Payoffs may be rare
  - How can “stage setters” be rewarded?

- **Classifier strength**
  - \( S(C,t) \) = strength of classifier \( C \) at time \( t \)
  - Only highest bidding classifiers post messages
  - \( B(C,t) = b \cdot R(C) \cdot S(C,t) \) = bid of classifier
  - \( b \) = probability that classifier posts (stochastic), \( R(C) \) = specificity of classifier

- **Complex economy**
  - Cost to place message: \( S(C,t+1) = S(C,t) - a \cdot B(C,t) \)
  - This cost is charged the consumer classifier, and is given to the supplier of the consumer
    - supplier = classifier sending messages that satisfy classifier’s condition
    - consumer = classifier with conditions satisfied by classifier’s message
  - A classifier will be profitable only if consumers are, otherwise consumers won’t be able to continue to bid
  - End payoff becomes distributed throughout agents that paved the way for the payoff
The Bucket-brigade Algorithm

When a classifier wins a competition it immediately
1) posts its message for use on the next time-step
2) pays its bid to its supplier(s) thereby reducing its strength.

In the diagram
C’ is first a consumer (of C) then a supplier (of C’).

(time of activation)  [t-1]    [t]    [t+1]
classifier

cond 1   cond 2   message

|tag x| |tag x| |tag y| |tag y|

(payment (time)  12 (t)  16 (t+1)

strength at t-1  100    120    160
strength at t    112    108    160
strength at t+1  112    124    144

Fig. 16. The bucket brigade algorithm
Genetic Programming (John Koza)

- Apply genetic algorithms to automatic program construction
  - Individuals = symbolic codes representing computer programs
    - Tree representations
    - Cross over by swapping tree structures
    - Lisp-like expressions: recursion and embedded expressions
  - Symbolic codes can be easily interpretable
  - Applied to function estimation, robot planning, sequence induction, rule generation, path finding, categorization, etc.

- Considerations
  - Need a domain where any randomly created code is interpretable
    - Math & logic are natural domains
  - Can incorporate simplicity into fitness function
  - Can create subroutines that are difficult to break-up
Crossover Applied to Tree Structures

Or (Not (D1), And (D0, D1))

Or (Or (D1, Not (D0)), And (Not (D0), Not (D1)))

The creation of XOR

The two crossover fragments are two sub-trees shown below:

These two crossover fragments correspond to the bold, underlined sub-expressions (sub-lists) in the two parental LISP S-expressions shown above. The two offspring resulting from crossover are shown below.

Note that the first offspring above is a perfect solution for the exclusive-or function, namely

(OR (AND (NOT D0) (NOT D1)) (AND D0 D1)).
Genetic Programming

The second best individual in the initial random population was

\[ x + [\log 2x + x] \times [\sin 2x + \sin x^2] \]

This individual has a raw fitness of 6.05. That is, the average distance between function and the curve for \(x^4 + x^3 + x^2 + x\) for the 20 points is about 0.3.

The best single individual in the population at generation 0 was the S-expression

\[ x + (\log 2x + x) \times (\sin 2x + \sin x^2) \]
The Santa Fe Trail Problem

Create a program for an ant to move on top of as many food squares as possible within a fixed number of steps, even though the trail has breaks in it.
The Santa Fe Trail Problem

Figure 1 Artificial Ant Solution