Genetic Algorithms
Evolution by Natural Selection

• Organisms inherit traits from parents

Computer (e.g., programs)

• Traits are inherited with some variation, via mutation and sexual recombination

• Due to competition for limited resources, the organisms best adapted to the environment tend to produce the most offspring.

• This way traits producing adapted individuals spread in the population
AN INTRODUCTORY ANALYSIS WITH APPLICATIONS TO
BIOLOGY, CONTROL, AND ARTIFICIAL INTELLIGENCE

ADAPTATION
IN
NATURAL
AND
ARTIFICIAL
SYSTEMS

JOHN H. HOLLAND
Some real-world uses of genetic algorithms
Designing parts of aircraft (GE and Boeing)

Optimization of the 787 Horizontal Stabilizer CFRP Composite Main Box

The -3 and -9 derivatives of Boeing's revolutionary 787 face a significant weight challenge due to very aggressive weight targets to achieve the desired efficiency. A major weight-trade study was launched to determine the optimal configuration and detail-sizing for the horizontal stabilizer CFRP co-cured main box. Using a genetic algorithm-based optimization solver (OptiStruct), various multi-spar configurations were optimized and evaluated. To determine the best path for further possible testing and allowable development, optimization was constrained to various buckling limits as well as to explore the addition of honeycomb core to both the skins and spars. The lowest additional development cost and risk, combined with a minimal weight, is chosen for each of the derivative models. Ultimately, the complete design of experiments highlights the development path toward the lightest possible composite main box structure.
Spacecraft antenna design (NASA)

Fig. 10. Sequence of evolved antennas leading up to antenna ST5-33.142.7.

http://idesign.ucsc.edu/papers/lohn_gptp05.pdf
Assembly line scheduling (John Deere Co.)
Automated drug discovery (several companies)

http://www.nature.com/nrd/journal/v7/n8/fig_tab/nrd2615_F4.html

Copyright © by Melanie Mitchell
Conference on Complex Systems,
         September, 2015
Fraud detection (credit cards, financial trading)

Expert Systems With Applications
2011 | 38 | 10 | 13057-13063

Detecting credit card fraud by genetic algorithm and scatter search
Ekrem Duman M. Hamdi Ozcelik
Generation of realistic computer animation
*(Lord of the Rings: The Return of the King and Troy)*

http://www.wired.com/wired/archive/12.01/stuntbots.html

Copyright © by Melanie Mitchell

Conference on Complex Systems,
September, 2015
Genetic Algorithm Example:

Evolving a Control Program for a Virtual “Robot”
Robby:  
The Virtual Soda Can Collecting Robot  
(Mitchell, 2009)

Herbert:  
The Soda Can Collecting Robot  
(Connell, Brooks, Ning, 1988)

http://cyberneticzoo.com/?p=5516
What Robby Can See and Do

Input:
Contents of North, South, East, West, Current

Possible actions:
Move N
Move S
Move E
Move W
Move random
Stay put
Try to pick up can

Rewards/Penalties (points):
Picks up can: 10
Tries to pick up can on empty site: -1
Crashes into wall: -5

Robby’s Score: Sum of rewards/penalties
Goal: Use a genetic algorithm to evolve a control program (i.e., strategy) for Robby.
What is a “strategy”?

**Strategy:** A set of rules that specifies an *action* for every possible *situation*.

**Possible Situations** = possible inputs to Robby

<table>
<thead>
<tr>
<th>North</th>
<th>South</th>
<th>East</th>
<th>West</th>
<th>Current Site</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
</tr>
<tr>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Can</td>
<td></td>
</tr>
</tbody>
</table>

North: 3 possibilities (Empty, Can, Wall) × South: 3 possibilities × East: 3 possibilities × West: 3 possibilities × Current Site: 3 possibilities = 3 × 3 × 3 × 3 × 3 = 243
### One Example Strategy

<table>
<thead>
<tr>
<th></th>
<th>North</th>
<th>South</th>
<th>East</th>
<th>West</th>
<th>Current Site</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>MoveNorth</td>
</tr>
<tr>
<td>2</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Can</td>
<td>MoveEast</td>
</tr>
<tr>
<td>3</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Wall</td>
<td>MoveRandom</td>
</tr>
<tr>
<td>4</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Can</td>
<td>Empty</td>
<td>PickUpCan</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>Wall</td>
<td>Wall</td>
<td>Wall</td>
<td>Wall</td>
<td>Wall</td>
<td>StayPut</td>
</tr>
</tbody>
</table>

**Question:** What will Robby’s score be after following this strategy for three time steps?
## Encoding a Strategy

<table>
<thead>
<tr>
<th>Situation</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>South</td>
</tr>
<tr>
<td>Empty</td>
<td>Empty</td>
</tr>
<tr>
<td>Empty</td>
<td>Empty</td>
</tr>
<tr>
<td>Empty</td>
<td>Empty</td>
</tr>
<tr>
<td>Empty</td>
<td>Empty</td>
</tr>
<tr>
<td>Wall</td>
<td>Empty</td>
</tr>
<tr>
<td>Wall</td>
<td>Wall</td>
</tr>
</tbody>
</table>
Encoding a Strategy

---

**Action**

1. MoveNorth
2. MoveEast
3. MoveRandom
4. PickUpCan
   ...  
243. StayPut

---

243 values

---

**Code:**
- MoveNorth = 0
- MoveSouth = 1
- MoveEast = 2
- MoveWest = 3
- StayPut = 4
- PickUpCan = 5
- MoveRandom = 6

---

Copyright © by Melanie Mitchell

Conference on Complex Systems, September, 2015
Question: How many possible strategies are there in our representation?

243 values

0 2 6 5 . . . 3 . . . 4

7 possible actions for each position:

$7 \times 7 \times 7 \times \ldots \times 7$

Goal: Have GA search intelligently in this vast space for a good strategy
Genetic algorithm for evolving strategies

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy, calculate fitness (average reward minus penalties earned on random environments)

3. The strategies pair up and create offspring via “sexual recombination” with random mutations — the fitter the parents, the more offspring they create.

4. Keep going back to step 2 until a good-enough strategy is found!
Random Initial Population

Individual 1:
23300323421630343530546006102562515114162260435654334066511514
15650220640642051006643216161521652022364433363346013326503000
40622050243165006111305146664232401245633345524126143441361020
150630642551654043264463156164510543665346310551646005164

Individual 2:
16411343121025360340361241431201104235462525304202044516433665
6103532215310513144062212061463143215461025652364422025340345
3050200562063402633100245341643015163121001221400664012665246
35165015412311313245330443321263455505314213064423311000

Individual 3:
2042334440241122613213645263246421220612212252660626144436125
325126640613353401534111110206164226653145522540234051155031302
22020065445125062206631426135532010000400031640130154160162006
134440626160505641421553133236021503355131253632642630551

...
Parent 1:

```
16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
30502005620634026331002453416430151631210012214400664012665246
351650154123113132453304433212634555005314213064423311000
```

Parent 2:

```
20423344402411226132136452632464212206122122252660626144436125
32512664061335340153411110206164226653145522540234051155031302
22020065445125062206631426135532010000400031640130154160162006
134440626160505641421553133236021503355131253632642630551
```

Mutation yields:

```
16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
30502005620634026331002453416430151631210012214400664012665246
351650154123113132453304433212634555005314213064423311000
```

Child:

```
16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
30502005620634026331002453416430151631210012214400664012665246
134440626160505641421553133236021503355131253632642630551
```
Maximum possible fitness $\approx 500$

- There are 100 squares total, and each environment starts out with about 50 cans.
- Each can is worth 10 points
My hand-designed strategy:

“If there is a can in the current site, pick it up.”

“Otherwise, if there is a can in one of the adjacent sites, move to that site.”

“Otherwise, choose a random direction to move in (avoiding walls).”

Average fitness of this strategy: 346 (out of max possible ≈ 500)

Average fitness of GA evolved strategy: 486 (out of max possible ≈ 500)
One Run of the Genetic Algorithm (C version)
Principles of Evolution Seen in Genetic Algorithms

- Natural selection works!
- Evolution proceeds via periods of stasis “punctuated” by periods of rapid innovation

- *Exaptation* is common
- Co-evolution speeds up innovation
- Dynamics and results of evolution are unpredictable and hard to analyze
• Netlogo Demo

• Robby code available at

http://web.cecs.pdx.edu/~mm/RobbyTheRobot/
Genetic Programming
(John Koza, 1990s)

John Koza
Genetic Programming (John Koza, 1990s)
Genetic Programming
(John Koza, 1990s)

Tree representation of programs

if (East=Can and North=Empty)
then MoveEast
else MoveSouth
A more complicated tree

```
ifelse
  and
    North=Empty
    East=Can
  ifelse
    ifelse
      MoveEast
      MoveWest
    Current=Can
    PickUpCan
  MoveSouth
  ifelse
    and
      not
      West=Can
      East=Can
      MoveEast
      MoveWest
```
Initial Population

Generate a population of random trees

Need to enforce some syntactic constraints, e.g., ifelse at root of tree, etc.
Fitness Calculation and Selection

Fitness:

Have Robby try out each strategy in a variety of environments; compute each strategy’s average score

Selection:

Fitter individuals create more offspring than less fit individuals
Crossover:
Exchange subtrees in corresponding branches to create child

Parents:

Child:
Mutation:
Replace a subtree by a randomly generated subtree

ifelse

and

East=Can

North=Empty

ifelse

South=Can

MoveNorth

ifelse

MoveSouth

MoveSouth
Mutation:
Replace a subtree by a randomly generated subtree

```
ifelse
  and
    East=Can
    North=Empty
ifelse
  MoveSouth
    South=Can
ifelse
  MoveNorth
    North=Can
ifelse
  MoveRandom
    North=Can
    StayPut
```
Genetic programming applied to Computer Graphics
(Karl Sims, 1993)

Karl Sims
Genetic programming applied to Computer Graphics (Karl Sims, 1993)

- GA individuals: trees representing equations that generate a color for each pixel coordinate
Each function returns an image (an array of pixel colors)

Left to right, top to bottom:

a. X
b. Y
c. (abs X)
d. (mod X (abs Y))
e. (and X Y)
f. (bw-noise .2 2)
g. (color-noise .1 2)
h. (grad-direction
   (bw-noise .15 2)
   0 0)
i. (warped-color-noise
   (* X .2) Y .1 2)
Each function returns an image (an array of pixel colors)
Some Results
(round (log (+ y (color-grad (round (+ (abs (round (log (+ y (color-grad (round (+ y (log (invert y) 15.5)) x) 3.1 1.86 #(0.95 0.7 0.59) 1.35)) 0.19) x)) (log (invert y) 15.5)) x) 3.1 1.9 #(0.95 0.7 0.35) 1.35)) 0.19) x)
• Website: http://www.karlsims.com/

• Applet: http://www.jhlabs.com/java/art.html
The viewers at this exhibit can observe a computer-simulated evolution in progress: an evolution of images. But in this evolution, the viewers are not just observers: they cause the evolution and direct its course.
A population of images is displayed by the computer on an arc of 16 video screens. The viewers determine which images will survive by standing on sensors in front of those they think are the most aesthetically interesting. The pictures that are not selected are removed and replaced by offspring from the surviving images. The new images are copies and combinations of their parents, but with various alterations. This is an artificial evolution in which the viewers themselves interactively determine the "fitness" of the pictures by choosing where they stand. As this cycle continues, the population of images can progress towards more and more interesting visual effects.
This interactive installation is an unusual collaboration between humans and machine: the humans supply decisions of visual aesthetics, and the computer supplies the mathematical ability for generating, mating, and mutating complex textures and patterns. The viewers are not required to understand the technical equations involved. The computer can only experiment at random with no sense of aesthetics — but the combination of human and machine abilities permits the creation of results that neither of the two could produce alone.