

# Evolutionary Algorithms in a Nutshell

*OR*

How I copied the comp.ai.genetic FAQ onto a bunch of slides at 5am this morning

Aniruddh Nath  
24 Sep 2006



# What are Evolutionary Algorithms?



- “A subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm.” - Wikipedia
- Wuh?
- A set of biologically inspired algorithms that work on the 'survival of the fittest' principle.

# Key Features



- Population
- Genetic operators (recombination, mutation)
- Reproduction
- Fitness function
- Exploration

# Types of Evolutionary Algorithms



- Genetic algorithms
- Evolutionary programming
- Evolution strategies
- Genetic programming
- Learning classifier systems

# Genetic Algorithms



- Search technique, often used for optimization problems.
- Population: usually a set of binary strings (“chromosomes”).
- Looking for a binary string that is exactly or approximately optimal, given a fitness function.

# Pseudocode (from c.a.genetic)

```
t := 0;
initpopulation P (t);
evaluate P (t);
while not done do
    t := t + 1;
    P' := selectparents P (t);
    recombine P' (t);
    mutate P' (t);
    evaluate P' (t);
    P := survive P,P' (t);
od
```

# Toy Example



---

Give me a number as close to '42' as possible.

Representation: unsigned binary strings.

Fitness function:  $f(x) = 42 - |42 - x|$

Completion condition:  $f(x) > 40$ .

# Toy Example (cont'd)



Init:  $P = \{010010, 110011, 100001\}$

Evaluate:

$$f(010010) = 18$$

$$f(110011) = 33$$

$$f(100001) = 33$$

# Generation 1

Select parents: 110011, 100001

Recombination: we're using one point crossover.  
Other methods are possible.

110		011	↖	110001
100		001	↙	100011

Mutation: 11**1**001, 10001**0**

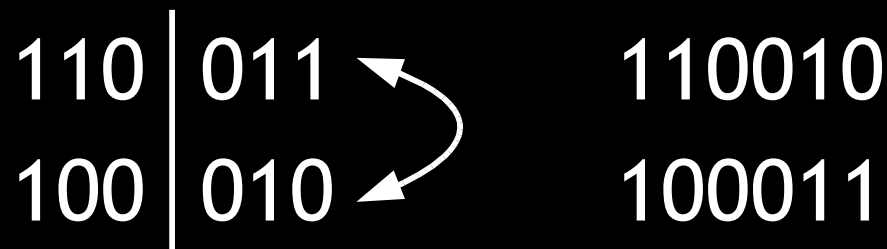
$P = \{110011, 100001, 111001, 100010\}$

Condition not met.

# Generation 2

Select parents: 110011, 100010

Recombination:



Mutation: 010010, 101011

$P = \{110011, 100010, 010010, 101011\}$

$f(101011) = 41 > 40$ . Done.

# Evolutionary Programming



- Similar idea, but not restricted to 'chromosome' structure.
- Solutions can have any structure; various mutations are possible based on how you define your solutions.
- Recombination tends not to play a role.

# Evolutionary Strategy



- Uses vector of reals rather than bit string.
- Mutation: add a random value to each element of the vector.
- Recombination: take the mean of each element of the parent vector:

$[1.0, 2.0, 3.0], [2.0, 1.0, 4.0] \longrightarrow [1.5, 1.5, 3.5]$

# Genetic Programming



---

- Solutions are actual programs, usually represented as parse trees.
- Recombination consists of programs exchanging subtrees.
- Mutation is generally not used.
- This is awesome.

# Learning Classifier Systems



---

- Population is a set of 'binary classifiers' (if-then rules about an environment).
- Uses reinforcement learning: the environment gives the agent a reward for choosing a set of rules; agent aims to maximize award.

# Problems with EAs



- Local maxima.
- Highly dependent on problem formulation.
- Not guaranteed to ever find a good solution (if you're really unlucky with your operators).

# Why bother?



- Yields surprisingly good results on optimization problems.
- Example: find a pretty good way to schedule jobs on a processor (minimize waiting time).
- Scheduling problems tend to be NP-hard; cannot calculate exact solution.
- EAs can also be used to find numbers close to 42.

# A Brief History of EAs



- Nils Aall Barricelli used EAs in 1954 to play a card game.
- The father of modern GAs is Prof. John Holland, at the University of Michigan.
- Prof. David Goldberg in the GE department worked with Prof. Holland, and now runs the Illinois Genetic Algorithms Laboratory (IlligAL).

# References/Further Reading



- Goldberg, David. *Genetic Algorithms in Search, Optimization, and Machine Learning*.
- comp.ai.genetic newsgroup.
- Wikipedia.
- Some website I found off Google:  
<http://neo.lcc.uma.es/cEA-web/ES.htm>