# $\lambda$ Lounge

## Evolutionary Algorithms

**Christopher Mark Gore**

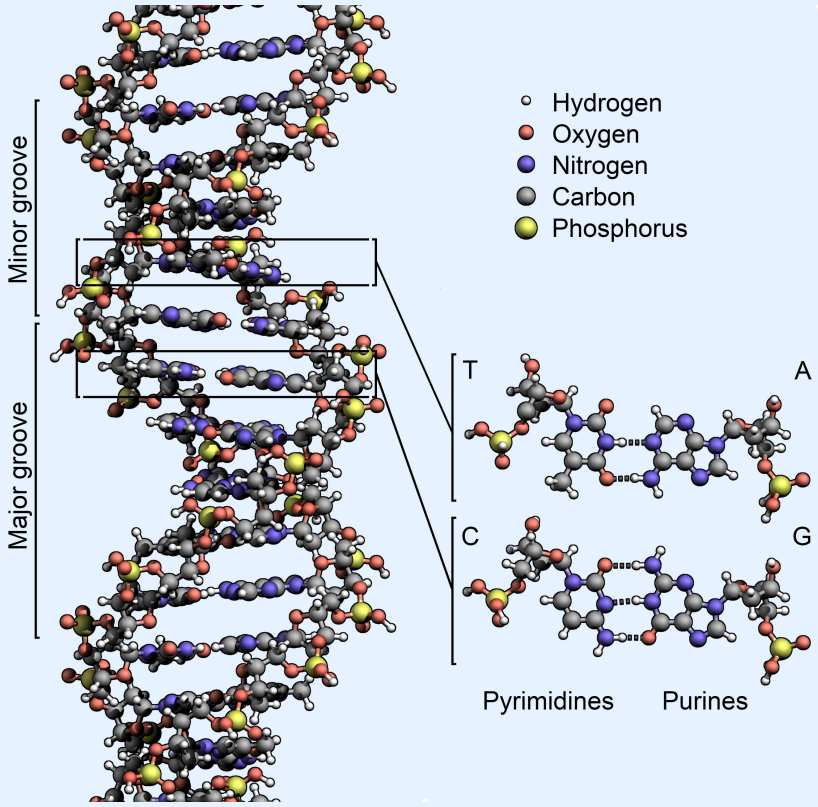
http://www.cgore.com

cgore@cgore.com

@cgore

**Thursday, March 1, 2012**

# Evolutionary Algorithms Are Cool!



## Why Use and Evolutionary Algorithm?

- The evaluation is very time-consuming.
- The problem is computationally difficult.
- You don't understand the problem.
- You are working with a real-world system.
- You just want to study evolutionary processes themselves.

## Where Are Evolutionary Algorithms Used?

- Artificial Intelligence.
- Financial modeling.
- Military applications.
- Complex engineering systems.
- Etc.

## The Evolutionary Algorithm

1. Population Initialization: generate a new population.
2. Fitness Evaluation: rate each member of the population.
3. Repeatedly:
   (a) Parent Selection: choose who to breed.
   (b) Recombination: cross the parents.
   (c) Mutation: applied to the offspring.
   (d) Fitness Evaluation: usually just the new offspring.
   (e) Survivor Selection: kill off the weak.

## Population Initialization

- Intelligently seed the population.
  Think Adam and Eve.
  **Pro:** you can start off the population somewhere useful.
  **Con:** you have to know where *"somewhere useful"* is.
- Randomly generate the population.
  This is usually easy and usually doesn't cost too many generations.
- Doing both at the same time isn't uncommon.

## Fitness Evaluation

We need to determine the fitness of our solutions.

- Minimum or maximum direct value.
- Simulation of an environment.
- Some fitness heuristic.
- Head-to-head competition.

## Parent Selection

- Breed the most fit.
- Don't be too selective: you'll get stuck on local optima.
- Wide-population breeding encourages exploration.
- Local breeding encourages optimization.

## Recombination

$$\begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} \oplus \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \epsilon \end{pmatrix} \rightarrow \begin{pmatrix} a \\ b \\ c \\ \delta \\ \epsilon \end{pmatrix}$$

## Mutation

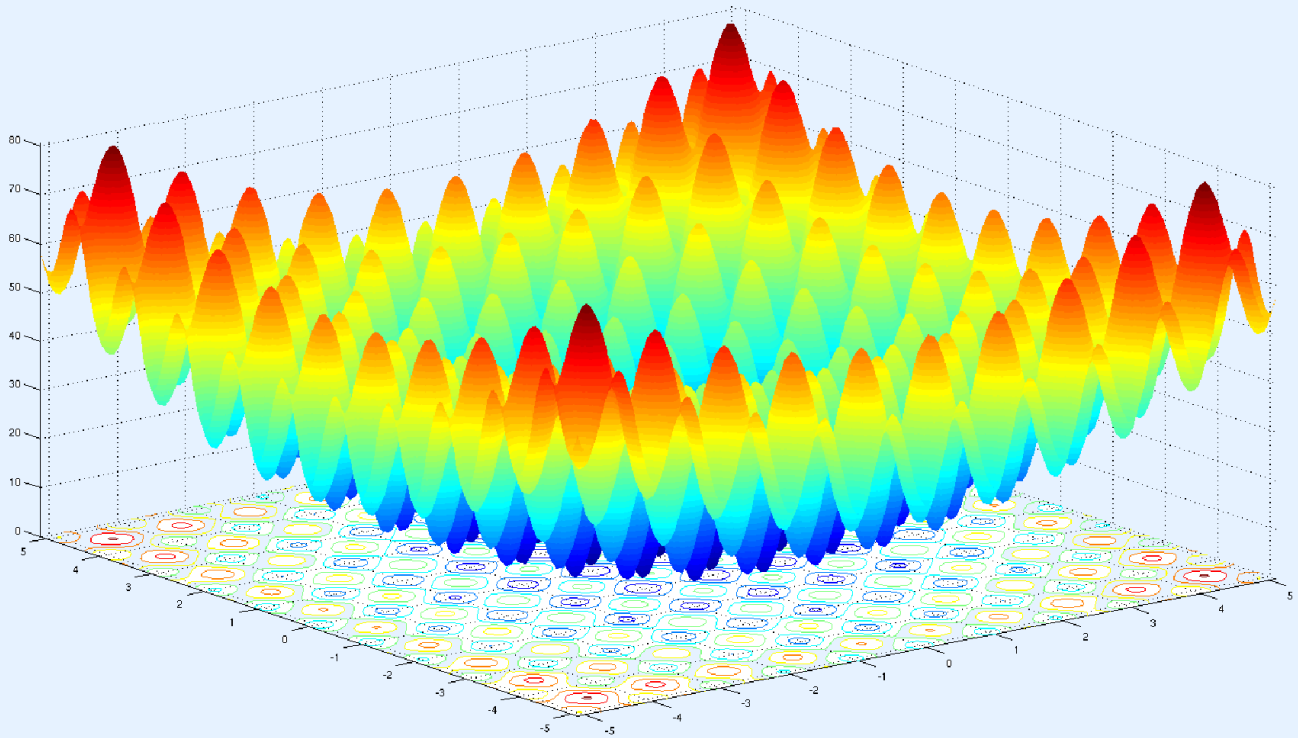
$$\begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} \rightarrow \begin{pmatrix} a \cdot 1.02 \\ b \\ c \cdot 0.975 \\ d \\ e \end{pmatrix}$$

## Survivor Selection

- Remove the least fit from the population:
  *"cull the herd."*
- Often just reduce the population to some maximum size.
- Don't be too selective:
  some population member may be useful for breeding later.

# The Rastrigin Function

$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[ x_i^2 - A\cos\left(2\pi x_i\right) \right]$$



## Let's Look at Some Code!

This is intentionally simple code.

- I want you to learn about evolutionary algorithms in this talk, not anything cool about Common Lisp.
- I want it to be an explanation, not an implementation.
- I want it to be as short and readable as possible.
- I want it to not use anything non-standard.

So in short, no, this isn't the best way to do things in the real world.

# *Questions?*