Understanding Genetic Algorithms
Part 1: the Problem

$X = (12.4, \text{"No-go"}, 22, \ldots)$
Understanding Genetic Algorithms
Part 2: Just Copy Nature

- Solution == DNA
  - e.g (12.4, “No-go”, 22,...)

- Fitness function
  - A method for determining how “good” a solution is
  - Can be a score, where higher is better

- Breeding
  - Combine DNA in different ways
    - E.g (12.4, “No-go”, 22,...) + (-3, “Go”, “9) == 2^x possible combinations

- Survival of the Fittest
Understanding Genetic Algorithms
Part 3: Inbreeding is Bad

\[ f(x) \]
Understanding Genetic Algorithms
Part 4: Or why the X-Men are the Best

- Need a method of ensuring Genetic Variability
- Mutations ensure that we’ll jump around the curve
Graph Guided Genetic Algorithms
Just cheating off of nature some more
Application: Genetic Engineering

ACGT - building blocks of DNA

“A nucleotide”

Amino Acids - building blocks of proteins

20 in all: (Tryptophan, Valine, Glutamine....)

Group of 3 nucleotides == 1 Amino Acid

$4^3 = 64$ combinations
There are redundancies + start and stop codons
Humans and Bacteria are Different

[Diagram showing DNA and bacterial cell with comparisons of CGT, CGC, CGA, and CGG frequencies]
Min-Max Estimations
The Problem

- 22 Possible Start/Stop/Amino Acids, 64 Codons, and DNA length of N
- For a given sequence, roughly 4 possible alternatives for a given Codon
- Search Space: $4^N$
Solution: Genetic Algorithms to solve Genetic Engineering Problems

- “DNA”: the specific Codon encodings which generate the same Protein
- Fitness Function: $\sum | \text{MinMax(human)} - \text{MinMax(SolutionInBacteria)} |$
- Breeding:
  - Zip Children: for each position, alternate between taking from parents
  - Skip Children: Zip Children where zip_num > 2
  - Random Children: randomly choose from parents
  - Half and Half: first half one parent, second half the other
  - ....
- Take top 10 each generation
How Graphs Made things Different

- *Graph Based Evolutionary Algorithms* by Bryden K.M. et al
- Take a graph and place a potential solution on each vertex
  - The only mating partners for that vertex are its neighbors
  - Choose from potential mates who to mate with
- Only replace parent if child is better than parent
The Graph I Used
Pseudocode

- Take target DNA strand, and create 20 Random variations, place one on each vertex in the graph
- For `i` in range(100):
  - For `v` in `graph.vertex()`:
    - `Children = []`
    - For `n` in `graph.neighbors(v)`:
      - `Children.add(breed(n,v,10))`
    - `Sort(children)`
    - If `children[0].score < v`:
      - `Graph.replace(v,children[0])`
- Return `sort(graph.vertex())[0]`
Results

Percent Min-Max Scores

New --- Orig

0  20  40  60  80  100  120  140  160
-20 -10  0   10  20  30  40  50  60  70  80  90
Future Direction

- Add more graphs and start timing, analyzing time and score based on different graphs and their properties
- Multi-thread the breeding process