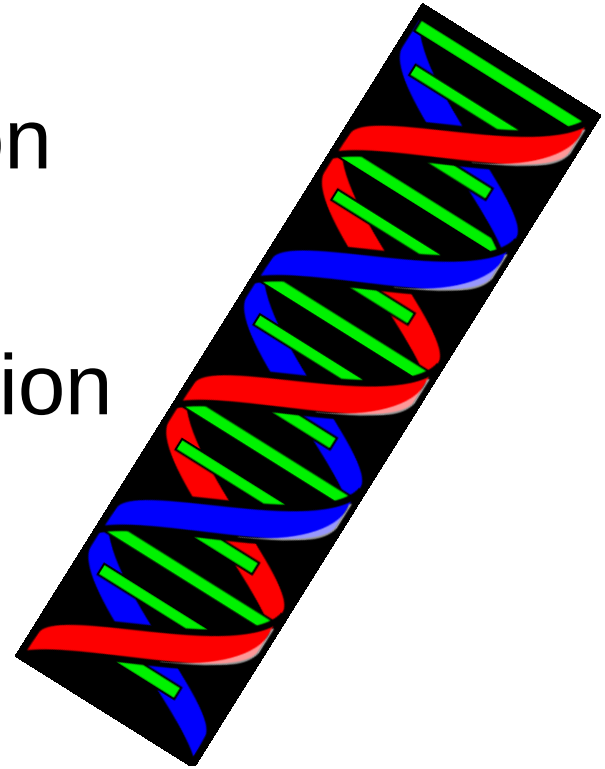
# Using Bio-Inspired Algorithms to Search for Solutions

- STEM Day Presentation
- October 21, 2017
- University of Mount Union



# Problem-Solving Strategies

- For many problems, multiple strategies exist for finding a solution
- Some of these strategies are more efficient than others
  - faster
  - use fewer computing resources
- Goal:  find the correct solution as efficiently as possible

# Example: I'm Thinking of a Number

- Think of a number between 1 and 1000….
- I'll try to guess your number
- You tell me if my guesses are too high or too low

- Three strategies can be employed …

# Strategies for Guessing a Number

- Strategy 1:  sequential values starting with one
  - Will this eventually guess the correct value?
  - Is there a better strategy?

# Strategies for Guessing a Number

- Strategy 2:  random guesses with no repeated guesses
  - Will this eventually work?
  - Is it better or worse than Strategy 1?

# Strategies for Guessing a Number

- Strategy 3:  pick middle value of the range of values we haven't eliminated yet
  - Will this work every time?
- It can be proven mathematically that no strategy is better than strategy 3 overall
- How fast is this?
  - 1000 values → max. 10 guesses
  - 1,000,000 values → max. 20 guesses!

# Consider another guessing game:

- I'm thinking of a 3-letter word or abbreviation
  - each of the three positions in my text string can be a letter or a space
- What is the best way to guess my text string?
- we can't do what we did in the number-guessing problem
  - There is no notion of “middle of what’s left”

# Generate random strings until we guess correctly…

- How long would it take to randomly generate a 3-letter text string?
- Define the alphabet to be capital letters and a blank space
- 27 possible characters for each letter position
- How likely is this approach to come up with the correct text string?

# Generate random strings until we guess correctly…

- Assume the target string is CAT
- 1st letter of phrase:
  - 27 possible choices
  - chance of getting a ‘C’:  1 in 27
- 2nd letter
  - 27 possible choices
  - chance of getting  ‘CA’:  1 in 27x27 = 1 in $27^2$
- 3rd letter
  - chance of getting ‘CAT’: 1 in $27^3$ = 1 in 19,683

# How Can Problems be Solved?

- There are many real-world problems for which there is no known “best” solution strategy
  - To solve the problem, there is no better solution than random guessing
- How are problems in the real world solved by living creatures?
  - By observing populations of organisms in nature, we can see a strategy that is better than random guessing

# Evolving Populations of Animals

- Consider a population of some kind of living creature (worms, ants, bears, …)
- The creatures must solve several problems in order to survive (find food, avoid predators, build nests, etc.)
- Some individuals in the population are better at these tasks than others
- solutions are innate (instincts)
  - built into the animal's genetic code

# Evolving Populations of Problem Solutions

- Represent each individual in a population by a genetic code sequence
- Rate each individual according to how well that creature solves the problem
- We can say that certain genetic code sequences are “better” (or more fit) at solving the problem than others
- This is the basic idea behind Evolutionary Computation

# Crucial Idea: Fitness of an Individual in the Population

- We need a way to compare individuals to see which ones solve the problem better than others
  - For our text-string guessing problem, let's define fitness to be the sum of how far away each letter of the guess is from the corresponding letter in the target string
- If our target = CAT
  - Fitness(DOG) = 1 + 14 + 13 = 28
  - Fitness(COW) = 0 + 14 + 3 = 17

# Evolutionary Algorithm

- Form a population of initial guesses, at random
- Evaluate the fitness of each member of the population
  - Must have some way of computing a fitness value that can be compared with the fitness of other guesses
- Evolve a series of generations of guesses, using fitter guesses as parents

# Evolutionary Algorithm

- Step 1: Generate a starting population at random
  - Draw 3 tiles from the buckets, with replacement
- Step 2: Form offspring
  - Choose parent(s)
  - Crossover mating
  - Mutation
- Step 3: Select individuals to survive into next generation, based on fitness

# Evolutionary Algorithm in Action

- Java program that uses EA to generate a target string…
  - Much better than pure random guessing
- Another problem: Visit all cities on a map and return to starting point
  - Called the Traveling Salesman Problem
    - fitness: total distance traveled (smaller is good)
  - Java program that uses EA to find a good solution to TSP…

# References


- Michalewicz, Zbigniew and David B. Fogel. How To Solve It: Modern Heuristics. Springer, 2002.
- Mitchell, Melanie. An Introduction to Genetic Algorithms. MIT Press, 1998.
- Olariu, Stephan and Albert Zomaya, ed. Handbook of Bioinspired Algorithms and Applications. Chapman & Hall, 2006.

# Using Bio-Inspired Algorithms to Search for Solutions

- STEM Day Presentation
- October 21, 2017
- University of Mount Union