

Lecture 7

Genetic Algorithms I

Lusi Li

Department of Computer Science
ODU

Review

- **Last Class**
 - Local Search
 - Hill-Climbing
 - Simulated Annealing
- **This Class**
 - Introduction to Genetic Algorithm
- **Next Class**
 - More on Genetic Algorithm

Biological Background – Natural Selection

- **The origin of species: “Preservation of favourable variations and rejection of unfavourable variations.” – Charles Darwin**
- **There are more individuals born than can survive, so there is a continuous struggle for life.**
- **Individuals with an advantage have a greater chance for survive: survival of the fittest.**

Evolutionary Computation

- **Natural Evolution**
 - Generating a population of individuals with increasing fitness
 - Increasing ability to survive and reproduce in a specific environment
- **Evolutionary Computation**
 - Simulate the natural evolution on a computer
 - Generate a set of solutions to a problem of increasing quality

Genetic Algorithms

- **Genetic Algorithms**
 - A class of search or optimization algorithms
 - Inspired by the biological evolution process
 - As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments.
 - By the 1975, the publication of the book *Adaptation in Natural and Artificial Systems*, by Holland and his students and colleagues.
- **Widely-used today in business, scientific and engineering circles**

What is GA

- A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to search or optimization problems.
- (GA)s are categorized as global search heuristics.
- (GA)s are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, selection, crossover (also called recombination), and mutation.

Terminology

- **Individual**
 - Any possible solution (or state) to a problem
- **Chromosome**
 - Representation of a solution
- **Gene**
 - Constituent entity of the chromosome
- **Population**
 - Set of individuals
- **Fitness Function**
 - Representation of how good a candidate solution is
- **Genetic operators**
 - Operators applied on chromosomes in order to create genetic variation (other chromosomes)

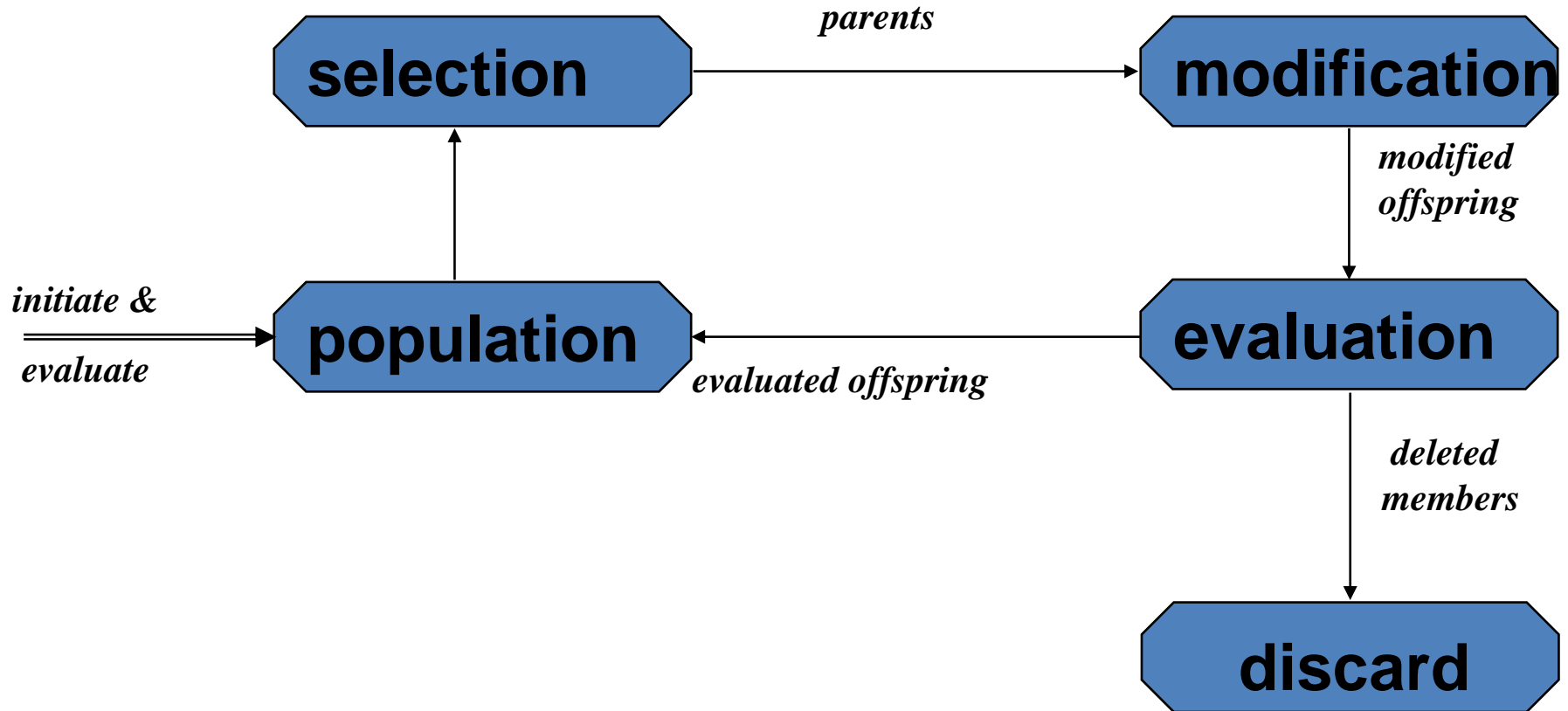
Basic Genetic Algorithms

- Start with n random solutions (the initial population)
- Repeatedly do the following:
 - Evaluate each of the solutions by the fitness function
 - Select a subset of the best solutions probabilistically
 - Use these solutions to generate a new population by
 - “crossover” with a rate
 - “mutation” with a rate
 - The new population is used in the next iteration
- Quit when you have a satisfactory solution (or you run out of time)

Terminology

- **Selection**
 - replicates the most successful solutions found in a population at a rate proportional to their relative quality
- **Crossover (Recombination)**
 - decomposes two distinct solutions (parents) and then randomly mixes their parts to form novel solutions (offspring)
- **Mutation**
 - randomly perturbs a candidate solution

The Evolutionary Cycle



Example: the MAXONE problem

- Suppose we want to maximize the number of ones in a string of k binary digits

Is it a trivial problem?

- It may seem so because we know the answer in advance
- However, we can think of it as maximizing the number of correct answers, each encoded by 1, to k yes/no difficult questions

Example (cont.)

- An individual is encoded (naturally) as a string of m binary digits
- The fitness f of a candidate solution to the MAXONE problem is the number of ones in its genetic code
- We start with a population of n random strings.
Suppose that $m = 10$ and $n = 6$

Example (Initialization)

- We toss a fair coin 60 times and get the following initial population:
- $n = 6$ individuals
 - each is encoded as a string of $m = 10$ binary digits

$$s_1 = 1111010101 \quad f(s_1) = 7$$

$$s_2 = 0111000101 \quad f(s_2) = 5$$

$$s_3 = 1110110101 \quad f(s_3) = 7$$

$$s_4 = 0100010011 \quad f(s_4) = 4$$

$$s_5 = 1110111101 \quad f(s_5) = 8$$

$$s_6 = 0100110000 \quad f(s_6) = 3$$

Example (Step1: Selection)

- Next, randomly (using a biased coin) select a subset of the individuals based on their fitness with the roulette wheel method:

$$f(s_1) = 7 \quad p(s_1) = 7/34$$

$$f(s_2) = 5 \quad p(s_2) = 5/34$$

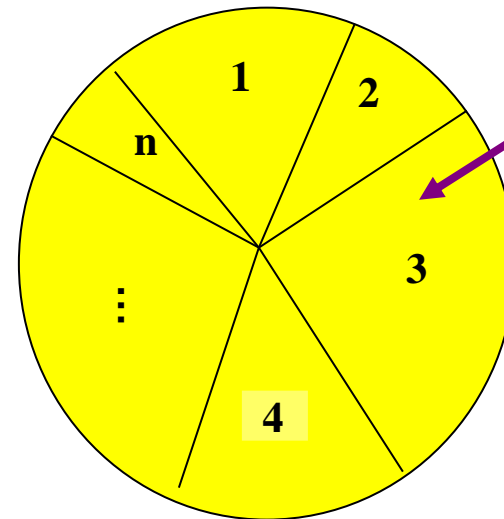
$$f(s_3) = 7 \quad p(s_3) = 7/34$$

$$f(s_4) = 4 \quad p(s_4) = 4/34$$

$$f(s_5) = 8 \quad p(s_5) = 8/34$$

$$f(s_6) = 3 \quad p(s_6) = 3/34$$

Individual i will have a probability to be chosen $\frac{f(i)}{\sum_i f(i)}$



Area is proportional to fitness value

We repeat the extraction as many times as the number of individuals we need to have the same parent population size (6 in our case).

Example (Selection Set)

Suppose that, after performing selection, we get the following population:

$$s_1' = 1111010101 \quad (s_1)$$

$$s_2' = 1110110101 \quad (s_3)$$

$$s_3' = 1110111101 \quad (s_5)$$

$$s_4' = 0111000101 \quad (s_2)$$

$$s_5' = 0100010011 \quad (s_4)$$

$$s_6' = 1110111101 \quad (s_5)$$

Example (Step2: Crossover)

- Next we mate strings for crossover.
 - For each couple we decide according to **crossover probability** (for instance 0.6) whether to actually perform crossover or not
 - Suppose that we decide to actually perform crossover only for couples (s1`, s2`) and (s5`, s6`). For each couple, we randomly extract a crossover point, for instance 2 for the first and 5 for the second

Example (crossover result)

Before crossover:

$s_1^{\text{'}} = 11\textcolor{red}{1010101}$ $s_2^{\text{'}} = 11\textcolor{blue}{10110101}$

Cross point = 2

$s_5^{\text{'}} = 01000\textcolor{red}{10011}$ $s_6^{\text{'}} = 11101\textcolor{blue}{11101}$

Cross point = 5

After crossover:

$s_1^{\text{'}} = 11\textcolor{blue}{10110101}$ $s_2^{\text{'}} = 11\textcolor{red}{1010101}$

$s_5^{\text{'}} = 01000\textcolor{blue}{11101}$ $s_6^{\text{'}} = 11101\textcolor{red}{10011}$

Example (Step3: Mutations)

The final step is to apply random mutation: for each bit that we are to copy to the new population we allow a **small probability of error** (for instance 0.1)

Before applying mutation:

$$s_1'' = 1110110101$$

$$s_2'' = 1111010101$$

$$s_3'' = 1110111101$$

$$s_4'' = 0111000101$$

$$s_5'' = 0100011101$$

$$s_6'' = 1110110011$$

After applying mutation:

$$s_1''' = 11101\mathbf{0}0101 \quad f(s_1''') = 6$$

$$s_2''' = 1111\mathbf{1}101\mathbf{00} \quad f(s_2''') = 7$$

$$s_3''' = 11101\mathbf{0}11\mathbf{11} \quad f(s_3''') = 8$$

$$s_4''' = 0111000101 \quad f(s_4''') = 5$$

$$s_5''' = 0100011101 \quad f(s_5''') = 5$$

$$s_6''' = 11101100\mathbf{0}1 \quad f(s_6''') = 6$$



The new population is formed and will be used in the next iteration.

Example (and now, iterate)

In one generation, the total population fitness changed from 34 to 37, thus improved by ~9%

At this point, we go through the same process all over again, until a stopping criterion is met

Components of a GA

A problem definition as input, and

- **Encoding principles** (gene, chromosome)
- **Initialization procedure** (creation)
- **Selection of parents** (reproduction)
- **Genetic operators** (crossover, mutation)
- **Evaluation function** (fitness to environment)
- **Termination condition**

Summary

- **Genetic Algorithm**
 - Crossover
 - Mutation
 - Application in MAXONE Problem

What I want you to do

- Review Course Slides