

Lecture 36

GENETIC ALGORITHM (1)

Outline

- *What is a Genetic Algorithm?*
 - An Example
- *Components of a Genetic Algorithm*
 - *Representation of gene*
 - *Selection Criteria*
 - *Reproduction Rules*
 - *Cross-over*
 - *Mutation*
- *Potential Applications of GA.*

What is a Genetic Algorithm?

- *Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics.*
- *They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search.*
- *In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure.*
- *While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance."*

- Genetic Algorithms in Search, Optimization & Machine Learning

by David E. Goldberg

What is a Genetic Algorithm?

Repeat

Evaluate current candidates

Develop new candidates via reproduction with modification
which replace least-fit former candidates

Until satisfied

Three Components:

- *Representation of candidate solutions (states) - Genes*
- *Selection criteria to evaluate the FITNESS of each gene*
- *Reproduction rules to generate new candidate solutions (genes) based on derivation from current solutions (cross-over breeding) and directed random search (mutation).*

A GA EXAMPLE

Objective - to find a binary string of length 5 with 4 1's.

Representation: binary string of length 5

Solution space: 5 feasible solutions among 2^5 solutions.

First step: randomly generate 5 candidates, and evaluate their fitness using the number of 1's in the string as a criterion.

00010 (eval: 1)

10001 (eval: 2)

10000 (eval: 1)

01011 (eval: 3)

10010 (eval: 2)

Population evaluation average: 1.8

GA EXAMPLE (2)

Second Step: generate new chromosomes

Modification methods:

- (a) crossover during which two genes interchange their chromosomes;*
- (b) inversion by flipping sub-string of the same gene; and*
- (c) mutation by randomly perturbation.*

Selectionist distribution: *Genes with higher fitness value has higher probability to produce off-springs!*

1 00010 (eval: 1)	2 10001 (eval: 2)	3 10001 (repeat)
4 10000 (eval: 1)	5 01011 (eval: 3)	6 01011 (repeat)
7 01011 (repeat)	8 10010 (eval: 2)	9 10010 (repeat)

Select pairs (indices from selectionist distribution): 1 & 4 @1, 4 & 5 @ 4, 9 & 7 @3, 8 & 6 @1, 7 & 5 @1

GENERATE NEW GENES

For example, crossover 1 (00010) and 4 (10000) at position 1 yields

00000 which evaluates 0! Other results are:

$4+5@4 = 10001$ (eval: 2)

$9+7@3 = 10011$ (eval: 3)

$8+6@1 = 11011$ (eval: 4)

$7+5@1 = 01011$ (eval: 3)

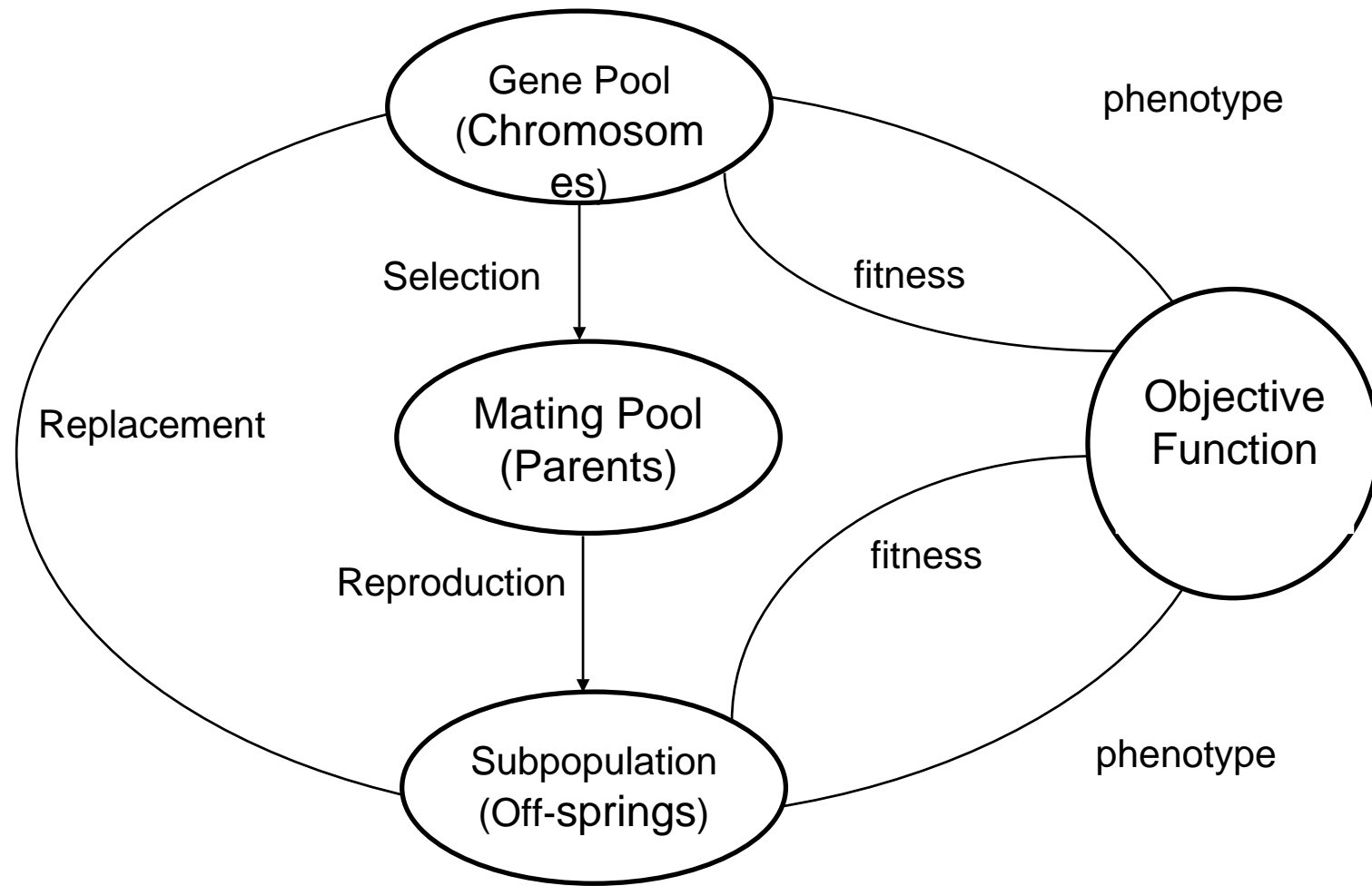
New population evaluation average: 2.4

Since $8 + 6$ produces a feasible solution, the iteration terminates, and the GA algorithm successfully found a solution.

GA Algorithm Overview

- GA is a random search algorithm which keeps a pool of candidate solutions (gene pool).
- Each solution is encoded in a binary string called a chromosome with each bit being a gene.
- Evaluate the fitness of a solution using a selection criteria.
- Generate new chromosomes by reproduction rules, including cross-over (mating), inversion, and mutation.
- Annihilate inferior (according to the result of evaluation using the selection criteria) genes, to make room for new genes.
- Adding new genes with high fitness values into gene pool.
- Evaluate termination criteria. If not yet satisfied, continue the search process.

GENETIC ALGORITHM CYCLE



GENE REPRESENTATION

Encoding is a key to the GA: Feature (knowledge) representation

Each chromosome is a vector of genes representing a trial solution.

Each gene can be a binary number, a real number or other symbols.

Bit-string encoding where each gene is a binary number is the most popular approach.

Other approaches: real number representation, order-based representation (good for graph coloring problem), embedded list (for factory scheduling), variable element lists (IC layout), and even LISP S-expressions.

SELECTION (FITNESS) CRITERIA

1. Windowing:

Let $v(i)$ = objective value of chromosome i , and c : a constant, then the fitness of chromosome i can be found as:

$$f(i) = c \pm [v(i) - v(w)]$$

where $v(w) < v(i)$ for all $i \neq w$.

2. Linear Normalization:

Rank objective values of chromosomes. Assign the best performed chromosome with fitness value $f(\text{best})$. Assign remaining i -th chromosome with fitness value

$$f(i) = f(\text{best}) - (i-1)d$$

PARENT SELECTION

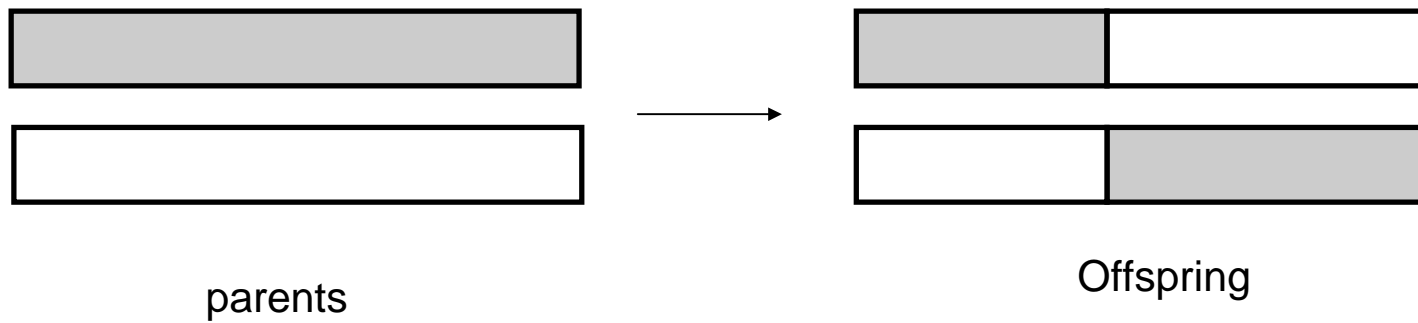
- *Emulate the survival-of-the-fittest mechanism in nature!*
- *In a Proportionate scheme where the growth rate of a chromosome with fitness value $f(x,t)$ is defined as $f(x,t)/F(t)$ where $F(t)$ is the average fitness of the population. An implementation is as follows:*

Roulette Wheel Parent Selection Algorithm

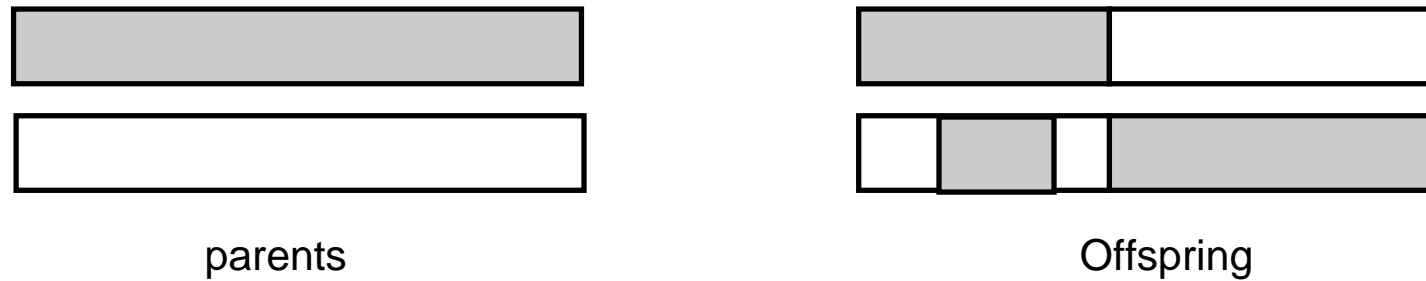
1. Sum the fitness of all population members; named as total fitness, n .
2. Generate a random number between 0 and n .
Return the first population member whose fitness added to the fitness of the preceding population members is greater than or equal to n

CROSS-OVER

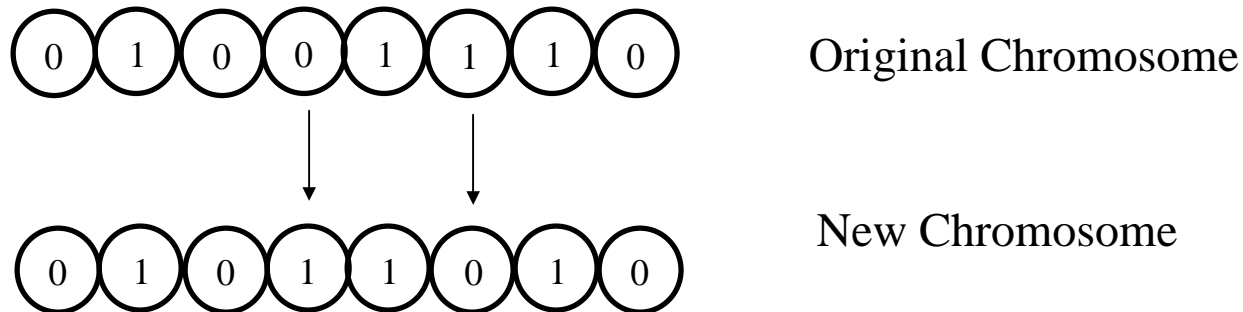
Single point crossover:



Multi-point crossover:



MUTATION



Mutation will take place with a small probability

For each bit in a bit stream, a probability test is performed. If passed, then one of two methods can be used:

Method 1. That bit is flipped (0 changes to 1, and vice versa)

Method 2. Randomly generate a bit. If the randomly generated bit is different from the original bit, the original bit is flipped.

REPLACEMENT STRATEGY

- Two strategies: Generation vs. Steady state replacement
- Generational Replacement - Copy the best or a few of the best chromosomes into a new generation. Generate remaining new chromosome to replace current generation.
- Steady State Replacement - Replace only the worst chromosomes with new chromosomes in each generation.

APPLICATIONS OF GENETIC ALGORITHMS

- Main Applications - Combinatorial Optimization:
 - VLSI physical design, layout optimization, routing placement
 - Job scheduling
- Signal Processing –
 - Time delayed estimation using FIR filter for sonar/radar
 - Speech coding
 - Wavelet coding
 - Image compression & recognition