Metaheuristic Optimization: Genetic Algorithms

AE 6310: Optimization for the Design of Engineered Systems

Spring 2017

Dr. Glenn Lightsey

Lecture Notes Developed By Dr. Brian German

Georgia Institute of Technology

Daniel Guggenheim School of Aerospace Engineering

Genetic Algorithms

Genetic algorithms are inspired by an analogy with natural selection ("survival of the fittest") in Darwin's theory of evolution.

Many designs are represented as *individuals* in a *population*.

Certain individuals are *selected* for *reproduction* based on their *fitness* relative to other individuals in the population.

After reproduction, some of the individuals are replaced by their offspring, and a new *generation* of the population is formed.

The invention of genetic algorithms is credited to John Holland.

Genetic Algorithms

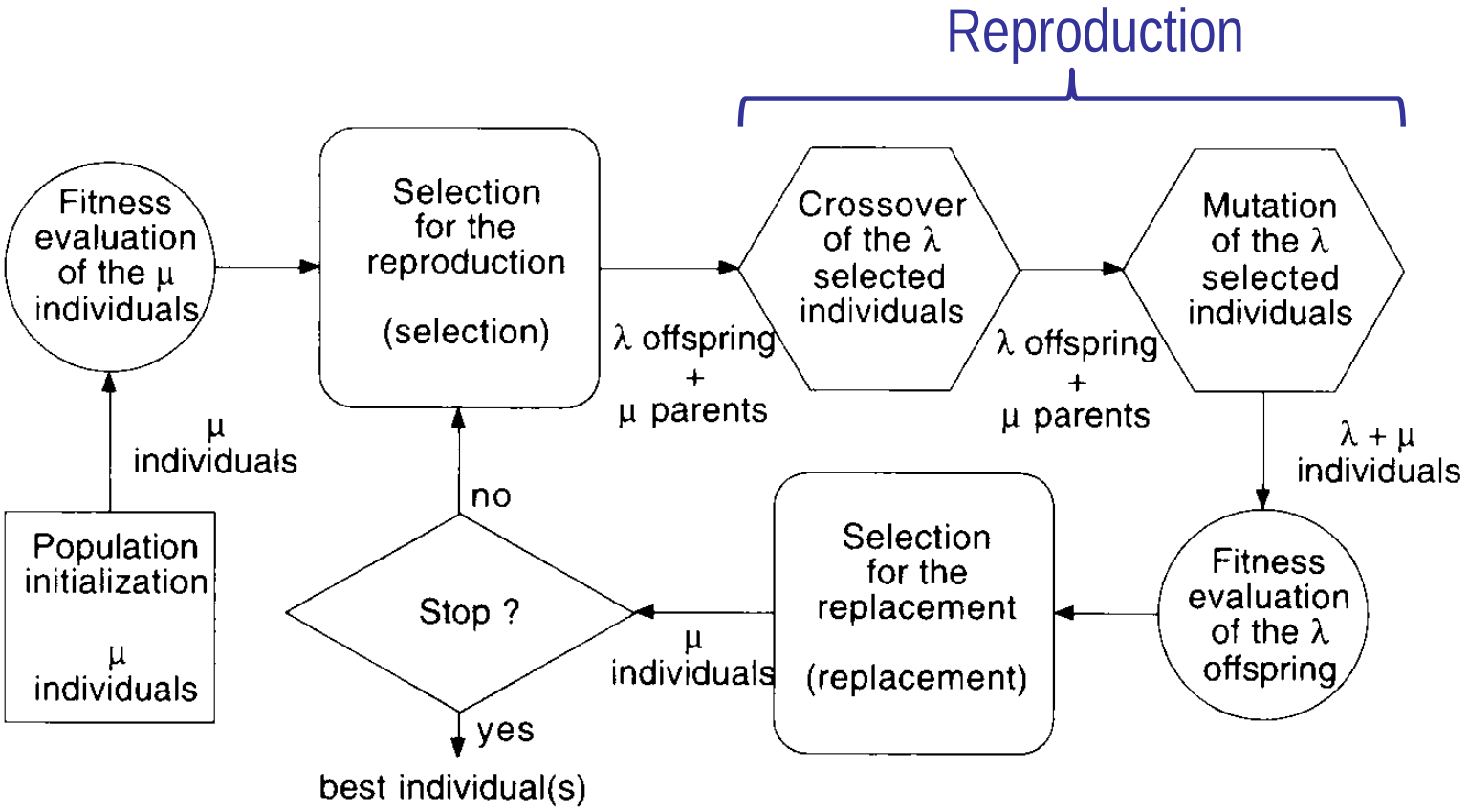


Fig. 3.1 from Dréo J., Pétrowski, A., Siarry, P., and Taillard, E., *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer 2006.

Genetic Algorithms

We will examine each of the steps in turn:

- Initialization
- Selection
- Crossover
- Mutation

} Reproduction

- Fitness evaluation
- Replacement

Initialization

To initialize the algorithm, a population of individuals must be generated.

If the problem involves continuous variables, these should be discretized over a desired range and to a desired resolution as binary numbers. It is preferable to represent the binary numbers in Gray code.

Concatenating the binary numbers for each variable forms a single string of n binary digits known as the ***chromosome*** for the individual.

The initial population of m individuals is generated by randomly sampling bits in each chromosome.

A rule of thumb is that $2n \leq m \leq 4n$.

Selection

There are two common forms for selecting individuals for reproduction:

- Proportional selection
- Tournament selection

We will look at both methods.

Selection

The general idea behind **proportional selection** can be stated as follows:

The expected number of selections of an individual for reproduction is proportional to its fitness.

Fitness is intended to be maximized, so the measure of fitness should monotonically increase with $f(\boldsymbol{x})$ if the optimization goal is to maximize, or monotonically increase with $-f(\boldsymbol{x})$ if the goal is to minimize.

Selection

A common form of proportional selection is called *roulette wheel selection*.

In roulette selection, each candidate is given a slice of an imaginary roulette wheel with the size of the slice proportional to its fitness. A random number is then generated to select a parent.

Selection

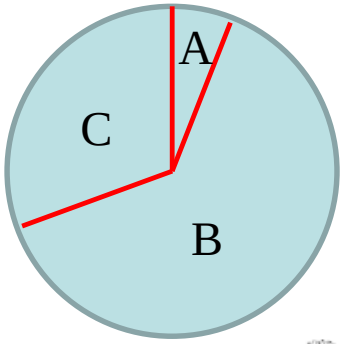
Example of roulette selection:

Candidates A, B and C have fitnesses 1, 10, and 5, respectively.

The roulette wheel would have the form A:[0,1], B:[1,11], C:[11,16].

Two random numbers between 0 and 16 are generated: 12 and 3.

Thus, the two parents are C and B.



Selection

Proportional selection encounters issues when there are very large or very small differences in the fitness function between individuals.

In the former case, a candidate with fitness orders of magnitude higher than the others would dominate the gene pool, diminishing the ability to explore.

In the latter case, if the population evolves to all be nearly optimal, the roulette wheel becomes nearly evenly spaced and the selection process resembles a random draw, hindering exploitation of the optimum.

Sometimes “monotone-preserving” transformations to the objective function are made to create a fitness function, e.g. log, exponential, or power law transformations, to address these issues.

Selection

However, in many cases, proportional selection gives each candidate a fair chance of becoming a parent.

This is important as we wish to retain some unfit candidates to promote exploration of the design space, yet narrow in on the best candidates to promote exploitation.

Selection

Tournament selection overcomes some of the difficulties of proportional selection.

As its name implies, tournament selection holds "tournaments" between different individuals to determine which will reproduce.

There are two general types of tournament selection:

- Deterministic tournament
- Stochastic tournament

Selection

Deterministic tournament selection

How it works:

- k individuals are chosen at random from the population. This is the *tournament size.*
- The k individuals participate in a "tournament" that compares their fitness values.
- The individual with the highest fitness "wins" the tournament and is selected for reproduction.

Selection

Stochastic tournament selection

How it works for a binary tournament ($k = 2$):

- 2 individuals are chosen at random from the population.
- Select the most fit individual with a probability $0.5 \leq p \leq 1$
- Choose the second most fit individual with probability $(1 - p)$

Can be generalized to arbitrary k by consideration of order statistics.

Selection

Since both proportional and tournament selection involve stochastic elements, there is no guarantee that the individuals in the population with the best fitness will be selected as parents or will persist into the next generation.

Elitism is the concept of copying the best individual(s) from one generation to the subsequent generation.

There are several strategies for implementing elitism, including copying the best individuals either before or after reproduction.

Crossover

Crossover is the genetic recombination of two parent individuals into two children.

It is the main mechanism of global search in GA.

After selecting two parents, a random number is generated to determine if crossover occurs. The probability is typically high (70% to 80%). If no crossover occurs, both parents simply become children (for most algorithms).

Crossover

There are three common types of crossover:

- One-point crossover
- Two-point crossover
- Uniform crossover

Crossover

One-point crossover

1. Given two parents with chromosome length l, a random integer k between 1 and $l - 1$ is generated. The value k indicates after which bit a "dividing line" should be drawn for the crossover.
2. Swap the relevant bits following k, i.e. bits $k + 1$ to l.

Crossover

Example of one-point crossover:

Parent A: 01110011 Parent B: 11001100 $(l = 8)$

Randomly generate an integer k between 1 and 7: 4

Parent A: $0111|0011$ Parent B: $1100|1100$

Swap the bits following the k^{th} bit:

Child A:01111100 Child B:11000011

Crossover

Two-point crossover is similar to one-point crossover except two random numbers between 1 and $l - 1$ are generated.

The first number represents the bit after which to start the swap, and the second number represents how many bits to swap. If the second number is large enough such that the end of the string is reached, continue counting from the beginning of the string.

Two-point crossover is less sensitive to the ordering of individual design variable strings in the overall chromosome.

Crossover

Example of two-point crossover:

Parent A: 00110011 Parent B: 11001100 $(l = 8)$

Randomly generate an integer between 1 and 7 (where to start): 7

Parent A: $0011001|1$ Parent B: $1100110|0$

Generate another integer between 1 & 7 (how many bits to swap): 3

Parent A: $\underline{00}|11001|\underline{1}$ Parent B: $\underline{11}|00110|\underline{0}$

Child A: 11110010 Child B: 00001101

Crossover

In uniform crossover, each bit is given a probability to swap. The probability is typically $p = 0.5$. For each bit, we generate a sample u from a uniform distribution over the range (0,1). If $u < p$, we swap the bits, $u > p$, we do not swap it.

Example of uniform crossover:

Parent A: 00110011 Parent B: 11001100 $(l = 8)$

Decide if each bit will swap: [Yes, No, No, No, No, Yes, No, No]

Parent A: $\underline{0}0110\underline{0}11$ Parent B: $\underline{1}1001\underline{0}00$

Child A: 10110011 Child B: 01001000

Mutation

After successive generations of selection and crossover, it will be likely that certain **alleles** will be eliminated from the gene pool (or may never have even occurred).

That is, a certain value at a certain position in the binary string is no longer in any of the candidates and can never appear again in any future children via crossover.

To maintain genetic diversity, mutation is added in the form of randomly flipping a few of the bits in the childrens' chromosomes.

It is important to implement mutation in a restrained manner or the GA will be little more than a glorified random search.

Mutation

There are several ways to implement mutation. Here are two:

- Similar to crossover, give each child a small probability to mutate (about 10%). If a child is chosen to mutate, randomly flip a user-defined number of bits in the child's chromosome. The locations within the string for the bit flips are chosen randomly.

- Every bit of every child's chromosome is given a small user-defined probability to flip (typically between 1% and 10%).

There are many other possible mutation strategies!

Replacement

Replacement is typically accomplished in one of the follow two ways:

- Replace all μ parents by all λ children to form the population for the next generation (this usually implies that $\lambda = \mu$ children are generated to preserve the population size over generations).
- Select the best μ individuals from all $\mu + \lambda$ parents and offspring to form the subsequent population.

Notice that the second method incorporates elitism.