Lecture Series on

Intelligent Control

Lecture 21
Evolutionary Computation and Genetic Algorithms

Kwang Y. Lee
Professor of Electrical & Computer Engineering
Baylor University
Waco, TX 76798, USA
Kwang_Y_Lee@baylor.edu

1

1

Evolutionary Computation

Computational Intelligence (CI):
- Evolutionary Computation (EC)
- Fuzzy Systems
- Artificial Neural Networks

Part of an even more complex universe, incorporating Artificial Life, Fractal Geometry, and other Complex Systems Sciences, which might someday be referred to as Natural Computation (NC).

2

2

Evolutionary Computation

- *Global Optimization algorithms* imitating certain principles of *created* nature have proved their usefulness in various domains of applications. Especially worth copying are those principles where nature has found "stable islands" in a "turbulent ocean" of solution possibilities.
- Such phenomena can be found in annealing processes, central nervous systems and biological evolutionary *hypothesis*; and have lead to the following optimization methods: Simulated Annealing (SA), Artificial Neural Networks (ANNs) and the field of Evolutionary Computation (EC).

3

3

Evolutionary Computation

- EC may currently be characterized by the following pathways:
  - Genetic Algorithms (GA),
  - Evolutionary Programming (EP),
  - Evolution Strategies (ES),
  - Classifier Systems (CFS),
  - Genetic Programming (GP), and
  - several other problem solving strategies,

Based upon biological observations dating back to Charles Darwin's *conjectures* in the 19th century - the means of natural selection and survival of the fittest, and *theories* of evolution. The inspired algorithms are thus termed Evolutionary Algorithms (EA).

4

4

Evolutionaty Algorithm

- *Evolutionary Algorithm* (EA):

  Computer-based problem solving systems, which use computational models of some of the *hypothetical* evolution mechanisms as key elements in their design and implementation.

- Common conceptual base:

  Simulate the evolution of individual structures via processes of *Selection*, *Mutation*, and *Reproduction*. The processes depend on the perceived Performance of the individual structures as defined by an *Environment*.

5

5

Evolutionary Algorithm

- *Population* of structures:

  Evolve according to rules of Selection and other operators, those are referred to as "search operators," (or Genetic Operators), such as Recombination and Mutation.

- Each *Individual* in the population receives a measure of its *Fitness* in the *Environment*.
- *Reproduction* focuses attention on high fitness individuals, thus exploiting the available fitness information.
- *Recombination* and *Mutation* perturb those individuals, providing general heuristics for Exploration.
- Although simple, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

6

6

Biological Basis

- Evolution is not a purposive or directed process:
  There is <u>no</u> evidence to support the assertion that the goal of evolution is to produce Mankind.
- The processes of nature seem to boil down to a haphazard Generation of biologically *diverse* organisms.
- Some of evolution is determined by natural Selection, or different Individuals *competing* for resources in the Environment. Some are better than others.
- Those that are better are *more likely* to survive and propagate their genetic material.

7

7

Biological Basis

- In nature, we see that the encoding for genetic information (Genome) is done in a way that admits asexual Reproduction.
- *Asexual reproduction* typically results in offspring that are genetically identical to the Parent. (Large numbers of organisms reproduce asexually; this includes most bacteria, which some biologists hold to be the most successful Species known).

8

8

Biological Basis

- *Sexual Reproduction* allows some shuffling of Chromosomes, producing offspring that contain a combination of information from each Parent. At the molecular level what occurs (wild oversimplification alert!) is that a pair of almost identical chromosomes bump into one another, exchange chunks of genetic information and drift apart.

- This is the Recombination operation, which is often referred to as *Crossover* because of the way that biologists have observed strands of chromosomes crossing over during the exchange.

9

9

Biological Basis

- Recombination happens in an Environment where the *Selection* of who gets to mate is largely a function of the Fitness of the Individual, i.e., how good the individual is at competing in its environment. Some "luck" (random effect) is usually involved too.
- Some EAs use a simple function of the *fitness* measure to select individuals (probabilistically) to undergo genetic operations such as Crossover or asexual Reproduction (the propagation of genetic material unaltered). This is *fitness-proportionate* Selection.

10

10

Biological Basis

- Other implementations use a model in which certain randomly selected individuals in a subgroup compete and the fittest is selected. This is called *Tournament Selection* and is the form of Selection we see in nature when stags rut to vie for the privilege of mating with a herd of hinds.
- Much EA research has assumed that the two processes that most contribute to Evolution are *Crossover* and *Fitness based Selection/reproduction*. As it turns out, there are mathematical proofs that indicate that the process of fitness proportionate Reproduction is, in fact, near optimal in some senses.

11

11

Biological Basis

- Evolution, by definition, absolutely requires diversity in order to work. In nature, an important source of diversity is *Mutation*. In an EA, a large amount of diversity is usually introduced at the start of the algorithm, by randomizing the Genes in the Population.
- The importance of Mutation, which introduces further diversity while the algorithm is running, therefore continues to be a matter of debate. Some refer to it as a background operator, simply replacing some of the original diversity which has been lost, while others view it as playing the dominant role in the evolutionary process.

12

12

Biological Basis

- It cannot be stressed too strongly that an evolutionary algorithm (as a Simulation of a genetic process) is *not* a random search for a solution to a problem (highly fit Individual). EAs use stochastic processes, but the result is distinctly non-random (better than random).

13

13

Evolutionary Algorithm

```
• Algorithm EA is
• t := 0;                          // start with an initial time
• initpopulation P (t);            // initialize a usually random population of individuals
• evaluate P (t);                  // evaluate fitness of all initial individuals in population
• while not done do                // test for termination criterion (time, fitness, etc.)
•       t := t + 1;                // increase the time counter
•       P' := selectparents P (t); // select sub-population for offspring production
•       recombine P' (t);          // recombine the "genes" of selected parents
•       mutate P' (t);             // perturb the mated population stochastically
•       evaluate P' (t);           // evaluate it's new fitness
•       P := survive P,P' (t);     // select the survivors from present fitness
• end EA.                          // done
```

14

14

Genetic Algorithm

- The Genetic Algorithm is a model of machine learning, which derives its behavior from a metaphor of some of the mechanisms of Evolutionary hypotheses.
- This is done by the creation within a machine of a Population of Individuals represented by *Chromosomes*, in essence a set of character strings that are analogous to the base-4 chromosomes that we see in our own DNA. The individuals in the population then go through a process of simulated "evolution".

15

15

Genetic Algorithm

- Applications:
  Multidimensional optimization problems in which the character string of the Chromosome can be used to encode the values for the different parameters being optimized.
- Implement this genetic model of computation by having arrays of bits or characters to represent the Chromosomes. Simple bit manipulation operations allow the implementation of Crossover, Mutation and other operations.

16

16

Genetic Algorithm

- Although a substantial amount of research has been performed on variable-length strings and other structures, the majority of work with Genetic Algorithms is focused on fixed-length character strings.
- We should focus on both this aspect of *fixed-length* and the need to *encode* the representation of the solution being sought as a character string, since these are crucial aspects that distinguish Genetic Programming, which does not have a fixed length representation and there is typically no encoding of the problem.

17

17

Genetic Algorithm

- When the Genetic Algorithm is implemented it is usually done in the following cycle:
  - Evaluate the Fitness of all of the Individuals in the Population.
  - Create a new population by performing operations such as *Crossover*, fitness-proportionate *Reproduction* and *Mutation* on the individuals whose fitness has just been measured.
  - Discard the old population and iterate using the new population.

18

18

Genetic Algorithm

- Implementation Model:
  One iteration of this loop is referred to as a *Generation*.
- The first (*initial*) Generation (generation 0) of this process operates on a Population of randomly generated Individuals.
- From there on, the genetic operations, in concert with the Fitness measure, operate to improve the population.

19

19

Genetic Algorithm

GA have certain characteristics that differentiate them from traditional *optimization* methods:

- GA *code* parameters in a bit string and not in the values of the parameters.
- GA search from a population of points, and not from a single point.
- GA use only the fitness function and don't need knowledge about derivatives or problem structure.
- GA use transition *probabilistic rules* (represented by the operators Selection, Crossover and Mutation), instead of deterministic rules.

20

20

Historical Perspectives

- In this point we refer to the main steps in the field of Genetic Algorithms since their birth in 1962 by the hand of J. Holland.
- Before that date some attempts were made in modeling genetic systems in computer systems (Barricelli, 1957, 1962; Fraser 1960, 1962, Martin and Cockerham, 1960). However these studies' fundamental objective was to understand some biological phenomena.
- John Holland and his students were the first to recognize the usefulness of using Genetic Operators in artificial adaptation problems.

21

21

Historical Perspectives

- Bagley in 1967 first mentioned the expression "Genetic Algorithm" and the first to present a practical application of this knowledge.
- In 1971, John Holland sets the basis for the theory behind the use of Genetic Algorithms: *The Schema Theorem*.
- In 1975 two important works were published:

  "Adaptation in Natural and Artificial Systems" of J. Holland, and

  "An Analysis of the behavior of a class of Genetic Adaptive Systems" of De Jong.

22

22

Historical Perspectives

- In 1989, Goldberg published

  "Genetic Algorithms in Search, Optimization and Machine Learning“, a reference book for Genetic Algorithms.
- In the present Genetic Algorithms are reaching their adult phase, being used in several applications in different fields, especially where conventional methods are not applicable.

23

23

Canonical Genetic Algorithm

- *Canonical algorithm:* Deals with three genetic operators (Selection, Crossover and Mutation) and linear, binary, fixed-size chromosomes.
- Canonical GA uses a fixed-size, non-overlapping population scheme and each new generation is created by the Selection operator and altered by Crossover and Mutation. The first population is generated at random.
- Also called *Simple Genetic Algorithm*.

24

24

Canonical Genetic Algorithm

Coding:

- Each chromosome represents a potential solution for the problem to solve and must be expressed in *binary* form.
- For instance, in the integer interval, I=[0,31], we could simply code x in binary base, using 5 bits (such as 10010 or 00101).
- If we have a set of binary variables, each variable will be represented by a *bit*.

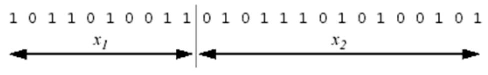
25

25

Canonical Genetic Algorithm

Coding:

- For a multivariable problem, each variable has to be coded in the chromosome.



- All of the search process takes place at the coding level.

26

26

Canonical Genetic Algorithm

Fitness:

- Each solution must be evaluated by the fitness function to produce a value.
- This objective function characterizes the problem to be solved and, playing the role of environment, establishes the basis for Selection.

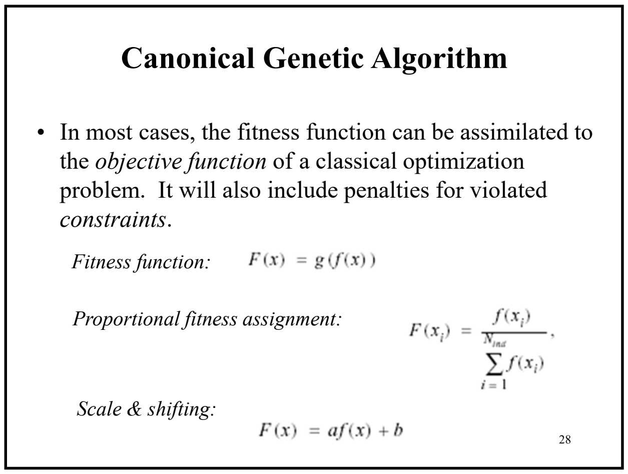
  e.g., If we want to maximize the function $f(x)=x^2$,

  11011 receives the fitness value $27^2=729$

  00111 receives the fitness value $7^2=49$
- The pair (chromosome, fitness) represents an *individual*.

27

27



Canonical Genetic Algorithm

- In most cases, the fitness function can be assimilated to the *objective function* of a classical optimization problem. It will also include penalties for violated *constraints*.

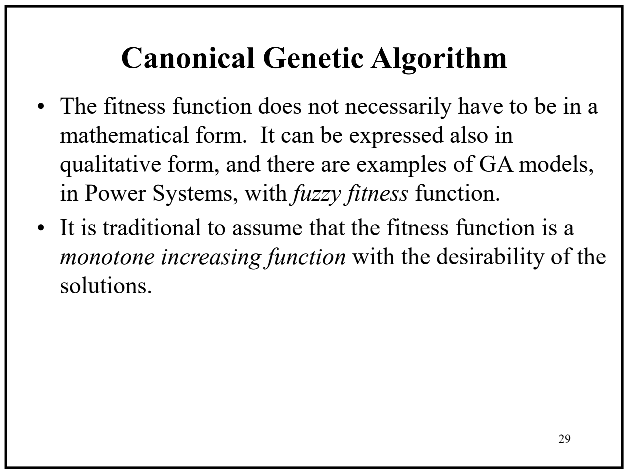
Fitness function: $F(x) = g(f(x))$

Proportional fitness assignment: $F(x_i) = \dfrac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)}$,

Scale & shifting: $F(x) = af(x) + b$

28

28

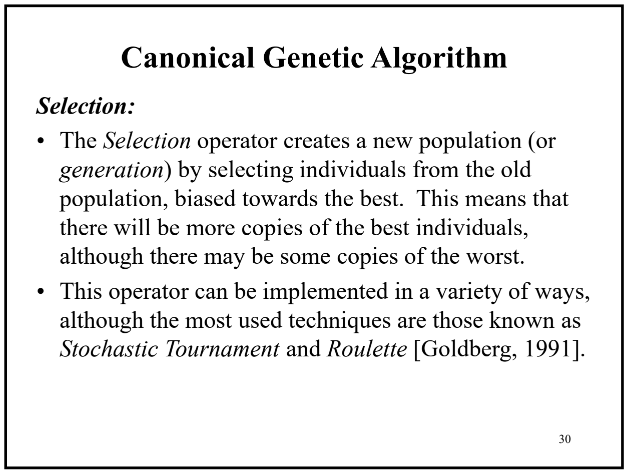


Canonical Genetic Algorithm

- The fitness function does not necessarily have to be in a mathematical form. It can be expressed also in qualitative form, and there are examples of GA models, in Power Systems, with *fuzzy fitness* function.
- It is traditional to assume that the fitness function is a *monotone increasing function* with the desirability of the solutions.

29

29



Canonical Genetic Algorithm

Selection:

- The *Selection* operator creates a new population (or *generation*) by selecting individuals from the old population, biased towards the best. This means that there will be more copies of the best individuals, although there may be some copies of the worst.
- This operator can be implemented in a variety of ways, although the most used techniques are those known as *Stochastic Tournament* and *Roulette* [Goldberg, 1991].

30

30

Canonical Genetic Algorithm

Stochastic Tournament -

- This implementation is suited to distributed implementations and is very simple: every time we want to select an individual for reproduction, we choose two, at random, and the best wins with some fixed probability, typically 0.8.
- This scheme can be enhanced by using more individuals on the competition [Goldberg, 1991] or even by considering evolving winning probability, eventually leading to *Boltzman Tournament* [Goldberg, 1991], generalizing the *Simulated Annealing* paradigm [Kirkpatrick, 1983].

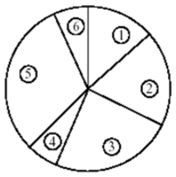
31

31

Canonical Genetic Algorithm

Roulette –

- In this process, the individuals of each generation are selected for survival into the next generation according to a probability value proportional to the ratio of individual fitness over total population fitness;



- this means that *on average* the next generation will receive copies of an individual in proportion to the importance of its fitness value.

32

32

Canonical Genetic Algorithm

We construct such a roulette wheel as follows:

Calculate the fitness value $f(x_i)$ for each chromosome x_i

Find the total fitness of the population:

$F = \text{sum}(f(x_i))$

Calculate the *probability of a selection* p_i for each chromosome:

$p_i = f(x_i)/F$

33

33

Canonical Genetic Algorithm

Calculate a cumulative probability q_i for each chromosome x_i

$q_i = \text{sum}\,(p_j;\ 1, \ldots, i)$

The selection process is based on spinning the *roulette wheel* n times; each time we select a single chromosome for a new population in the following way:

Generate a random number r from the range [0, 1].

If $r<q_1$ then select the first chromosome (x_1); otherwise select the i-th chromosome x_i ($2<i\leq n$) such that $q_{i-1}<r\leq q_i$.

34

34

Canonical Genetic Algorithm

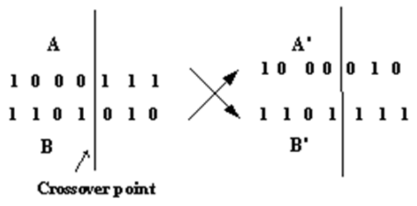
Crossover -

- The recombination operator used in the Canonical Genetic Algorithm is called Single Point Crossover. Individuals are paired at random with a high probability that Crossover will take place.
- In affirmative case, a Crossover point is selected at random and, say, the rightmost segments of each individual are exchanged to produce two offsprings as illustrated in the next figure, where two 7-bit chromosomes **A** and **B** exchange parts (the Crossover point is p=4) resulting in the chromosomes **A'** and **B'**:

35

35

Canonical Genetic Algorithm

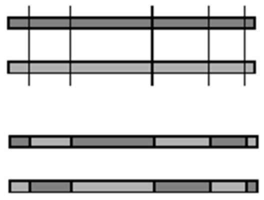


Crossover operation.

36

36

Canonical Genetic Algorithm



Multi-point Crossover operation

37

37

Canonical Genetic Algorithm

```
P1   = 1 0 1 1 0 0 0 1 1 1
P2   = 0 0 0 1 1 1 1 0 0 0

Mask = 0 0 1 1 0 0 1 1 0 0

O1   = 0 0 1 1 1 1 0 1 0 0
O2   = 1 0 0 1 0 0 1 0 1 1
```

Uniform Crossover operation

38

38

Canonical Genetic Algorithm

Mutation -

- In the Canonical Genetic Algorithm Mutation consists of simply flipping each individual bit with a very low probability (A typical value would be $P_m = 0.001$). This background operator is used to ensure that the probability of searching a particular subspace of the problem space is never zero, thereby tending to inhibit the possibility of ending the search at a local, rather than a global optimum.

39

39

Canonical Genetic Algorithm

Crossover:

$P_1 = 1\ 0\ 0\ 1\ 0\quad 1\ 1\ 0,$

$P_2 = 1\ 0\ 1\ 1\ 1\quad 0\ 0\ 0.$

$O_1 = 1\ 0\ 0\ 1\ 0\quad 0\ 0\ 0,$

$O_2 = 1\ 0\ 1\ 1\ 1\quad 1\ 1\ 0.$

Mutation:

$O_{1m} = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0.$

40

40

Canonical Genetic Algorithm

Parameters-

- Like other optimization methods, GA have certain parameters like, for example:
  - Population size;
  - Genetic Operations Probabilities;
  - Number of individuals involved in the Selection procedure, etc.
- These parameters must be selected with maximum care, for the performance of GA depends largely on the values used. Normally, it's recommended to use a relatively *low* population number, *high* Crossover and *low* Mutation probabilities.

41

41

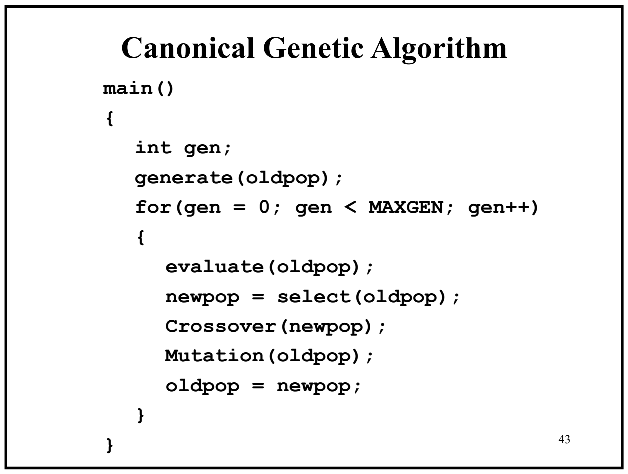
Canonical Genetic Algorithm

How GA Works -

- A canonical GA is a very simple process: we first generate a random initial population, evaluate it and start creating new populations by applying genetic operators. This high-level behavior can be depicted on the following piece of pseudo-C:

42

42



Canonical Genetic Algorithm

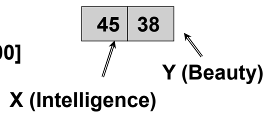
```
main()
{
   int gen;
   generate(oldpop);
   for(gen = 0; gen < MAXGEN; gen++)
   {
      evaluate(oldpop);
      newpop = select(oldpop);
      Crossover(newpop);
      Mutation(oldpop);
      oldpop = newpop;
   }
}
```

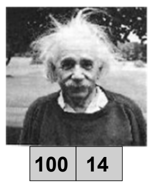
43

43

The importance of recombination

Max the success in life f
$f(x,y) = x^2 + y^2$ $x,y \in [0,100]$

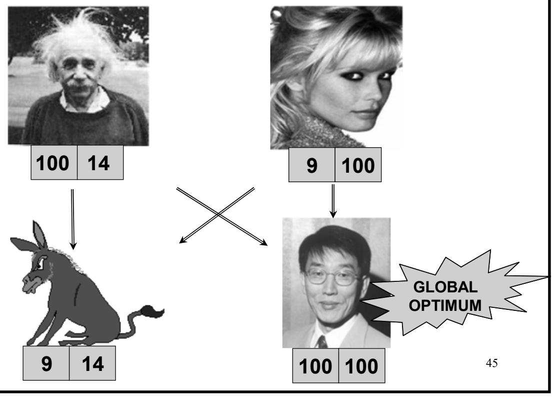




44

44

The importance of recombination



45

45