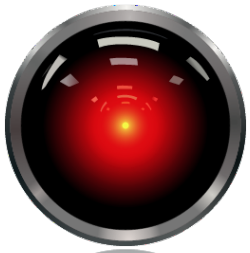


INTELLIGENT COMPUTING

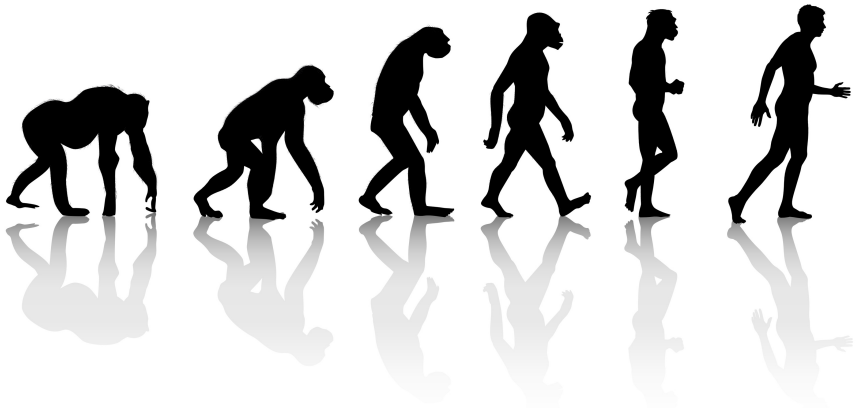
(COMPUTATIONAL INTELLIGENCE)
Evolutionary Computing - Genetic Algorithms

S. Sheridan

WHAT IS EVOLUTIONARY COMPUTING?

- Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution.
- This lecture considers learning algorithms fashioned after the process underlying evolution: shaping a population of individuals through the survival of its most fit members through natural selection.
- Evolutionary computing techniques mostly involve optimisation algorithms. Broadly speaking, the field includes but is not limited too: Evolutionary algorithms, Genetic algorithms, Genetic programming, Evolutionary programming, Differential evolution, Swarm intelligence, Ant colony optimization, Particle swarm optimization, Artificial life and Artificial immune systems.

WHAT IS EVOLUTIONARY COMPUTING?



Primates diverged from other mammals about 85 million years ago in the Late Cretaceous period.

WHAT IS EVOLUTIONARY COMPUTING?

- Emergent models of learning simulate nature's most elegant and powerful form of adaptation: the evolution of plant and animal life forms.
- Charles Darwin saw "..no limit this power of slowly and beautifully adapting each form to the most complex relations of life."
- Through this simple process of introducing variations into successive generations and selectively eliminating less fit individuals, adaptations of increasing capability and diversity emerge in a population.
- Evolution and emergence can occur in populations of *embodied* individuals, whose actions affect others and that, in turn, are affected by others.

WHY MODEL BIOLOGICAL EVOLUTION?

- Many computational problems require searching through a huge number of possibilities for solutions. Such problems can often benefit from an effective use of parallelism, in which many different possibilities are explored simultaneously in an efficient way.
- What is needed is both computational parallelism (i.e. many processors evaluating sequences at the same time) and an intelligent strategy for choosing the next set of sequences to evaluate.
- Many computational problems require a computer program to be adaptive and continue to perform well in a changing environment. Other problems require computer programs to be innovative -- to construct something truly new and original.
- Many computational problems require complex solutions that are difficult to program by hand.

WHY MODEL BIOLOGICAL EVOLUTION?

- Biological evolution is an appealing source of inspiration for addressing these problems. Evolution is, in effect, a method of searching among an enormous number of possibilities for solutions.
- In biology the enormous set of possibilities is the set of possible genetic sequences, and the desired solutions are highly formed organisms. Organisms well able to survive and reproduce in their environment.
- Evolution can also be seen as a method for designing innovative solutions to complex problems. For example, the mammalian immune system is a marvellous evolved solution to the problem of germs invading the body.

WHY MODEL BIOLOGICAL EVOLUTION?

- Seen in this light, the mechanisms of evolution can inspire computational search methods. Of course the fitness of biological organism depends on many factors -- for example, how well it can compete with or cooperate with the other organisms around it.
- The fitness criteria continually change as creatures evolve, so evolution is searching a constantly changing set of possibilities. Searching for solutions in the face of changing conditions is precisely what is required for adaptive computer programs.
- Furthermore, evolution is a massively parallel search method: rather than working on one species at a time, evolution tests and changes millions of species in parallel.
- Finally viewed from a high-level the "rules" of evolution are remarkably simple. Yet these simple rules are thought to be responsible, in a large part, for the extraordinary variety and complexity of life on our planet.

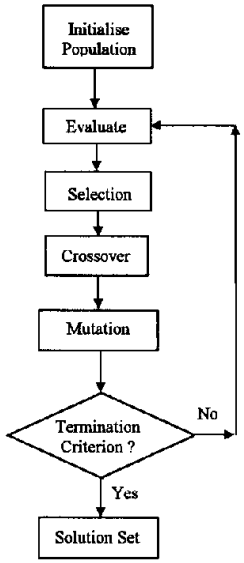
WHAT IS A GENETIC ALGORITHM?

- In the field of computer science, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems.
- Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.
- Genetic algorithms find application in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics, pharmacometrics and other fields.

WHAT IS A GENETIC ALGORITHM?

- When a genetic algorithm is used to solve a problem; it has three distinct stages:
  - *First the individual potential solutions for a problem domain are encoded in a way that supports evolutionary operators. One common way to do this is to represent the potential solutions as bit strings.*
  - *Secondly, mating and mutation algorithms, analogous to the reproduction of biological life forms, produce new generations of individuals that recombine features of their parents.*
  - *Finally a fitness function judges which individuals are the "best" life forms, that is, most appropriate for the eventual solution of the problem.*
- These individuals are favoured in survival and reproduction, shaping the next generation of potential solutions.

WHAT IS A GENETIC ALGORITHM?



WHAT IS A GENETIC ALGORITHM?

- Let P(t) define a population of candidate solutions, xi, at time t:

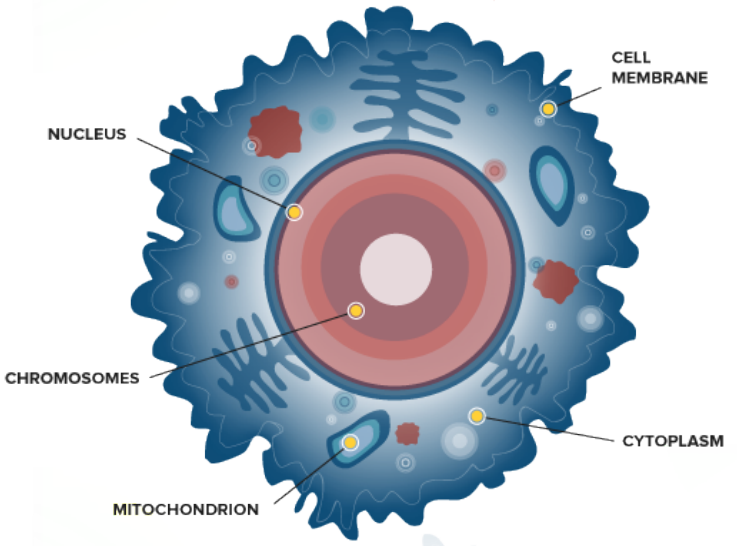
$$P(t) = \{x^t_1, x^t_2, x^t_3, x^t_4, \ldots x^t_n\}$$

```
Procedure genetic_algorithm
begin
  set time t = 0;
  initialise the population P(t);
  while the termination condition is not met do
     begin
        evaluate fitness of each member of the population P(t);
        select most fit members from the population P(t);
        produce the offspring of these pairs using genetic operators;
        replace the weakest candidates of P(t) with these offspring;
        set time t = t + 1;
     end
end
```


EVALUATION / FITNESS FUNCTION

- A fitness function is a particular type of objective function that is used to summarise, as a single figure of merit, how close a given design solution (chromosome) is to achieving the set aims.
- After each evolution step, the idea is to delete the 'n' worst chromosomes, and to breed 'n' new ones from the best chromosomes Each chromosome, therefore, needs to be awarded a figure of merit, to indicate how close it came to meeting the overall specification, and this is generated by applying the fitness function to test the result obtained from that chromosome.
- It is the human designer who has to design the fitness function. If this is designed badly, the algorithm will either converge on an inappropriate solution, or will have difficulty converging at all. **NOTE:** We could use fuzzy logic to evaluate each chromosome.

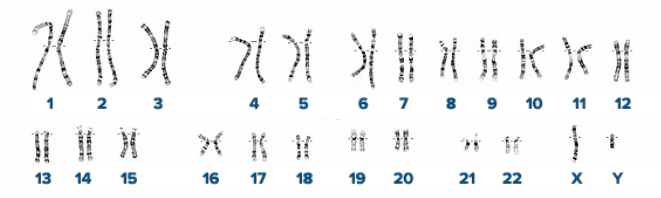
SOME BIOLOGICAL TERMINOLOGY

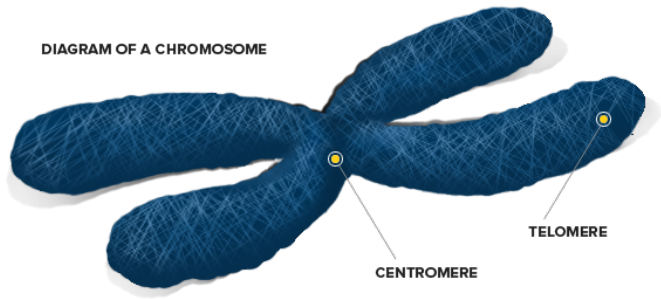


SOME BIOLOGICAL TERMINOLOGY

- All living organisms consist of cells, and each cell contains the same set of one or more **chromosomes** -- string of DNA -- that serve as a "blueprint" for the organism.
- A chromosome can be conceptually divided into **genes** -- functional blocks of DNA, each of which encodes a particular protein. Very roughly one can think a gene as encoding a trait, such as eye colour.
- The different possible "settings" for a trait (e.g., brown, blue, hazel) are called **alleles**.
- Each gene is located at a particular **locus** (position) on the chromosome.
- Many organisms have multiple chromosomes on each cell. The complete collection of genetic material is called the organisms **genome**.

SOME BIOLOGICAL TERMINOLOGY





SOME BIOLOGICAL TERMINOLOGY

- During reproduction, **recombination** (or **crossover**) occurs: in each parent, genes are exchanged between each pair of chromosomes to form a gamete (a single chromosome), and then gametes from the two parents pair up to create a full set of chromosomes.
- Offspring are subject to **mutation**, in which single nucleotides (elementary bits of DNA) are changed from parent to offspring, the changes are often resulting from copying errors.
- The **fitness** of an organism is typically defined as the probability that the organism will live to reproduce (**viability**) or as a function of the number of offspring the organism has (**fertility**).

SOME BIOLOGICAL TERMINOLOGY

- In genetic algorithms, the term chromosome typically refers to a candidate solution to a problem, often encoded as a bit string. The "genes" are either single bits or short blocks of adjacent bits that encode a particular element of the candidate solution.
- An allele in bit string is either 0 or 1; for larger alphabets more alleles are possible at each locus.
- Crossover typically consists of exchanging genetic material between two parents.
- Mutation consists of flipping the bit at a randomly chosen locus with randomly chosen new symbol.
- The fitness of a chromosome is usually calculated based on some kind of fitness function that is suitable for the problem at hand.

ENCODING CHROMOSOMES

- The chromosome should in some way contain information about solution which it represents. The most common approach is to encode chromosomes as binary strings.

Chromosome 1	1101100100110110
Chromosome 2	1100111000011110

- Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution. Or the whole string can represent a number - this approach is used in the example covered later in this lecture.
- Of course, there are many other ways of encoding. The encoding approach will depend on the problem being solved.

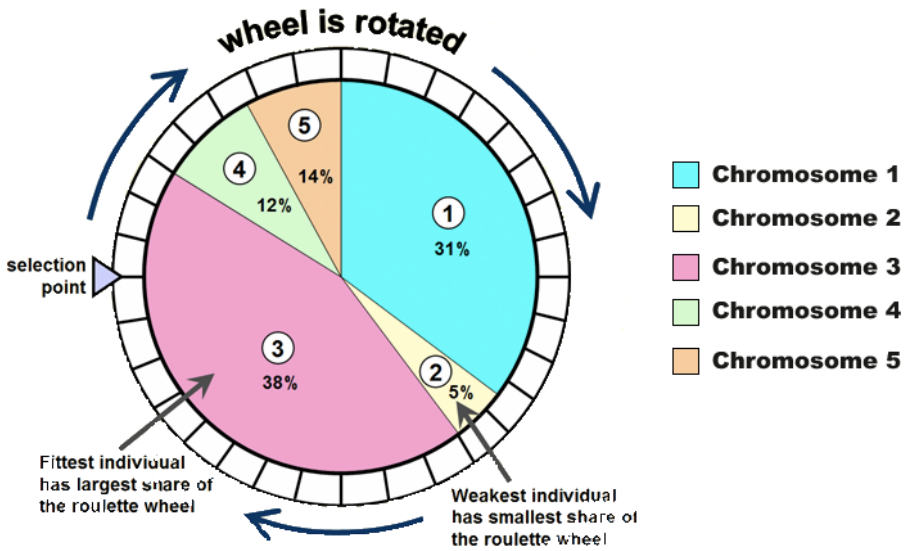
CHROMOSOME SELECTION PROCESS

- As you already know from previous slides, chromosomes are selected from the population to be parents. So how are these parent chromosomes selected?
- According to Darwin's evolution theory the best ones should survive and create new offspring. There are many different ways to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.
- For the purposes of our brief investigation of genetic algorithms we will concentrate on only one method, roulette wheel selection.

ROULETTE WHEEL SELECTION

- Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. This can be achieved by using a roulette wheel to select parent chromosomes.
- Unlike a real roulette wheel where all slots are the same size, the roulette wheel used to select chromosomes, sizes its slots based on the fitness of each chromosome. This increases the probability of a chromosome with a high degree of fitness being selected for crossover/recombination.

ROULETTE WHEEL SELECTION



ELITISM

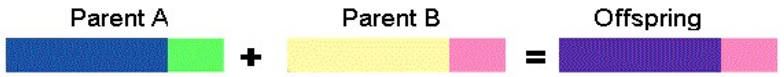
- When creating a new population by crossover and mutation, there is a big chance, that we may loose the best chromosome.
- Elitism is the name of a method, which first copies the best chromosome (or a few best chromosomes) to the new population.
- Elitism can very rapidly increase performance of GA, because it prevents losing the best solution.

BINARY CROSSOVER AND RECOMBINATION

- Once an encoding approach has been chosen we must then decide on how the genetic operators such as crossover and mutation will be implemented for the given encoding.
- Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way to do this is to choose a crossover point at random and swap bits from each parent using this point as the centre.

SINGLE POINT CROSSOVER

- Single point crossover - one crossover point is selected at random, a binary string from the beginning of chromosome A to the crossover point, is copied to the same position in the new offspring, the rest is copied from the second parent.



11001011+11011111 = 11001111

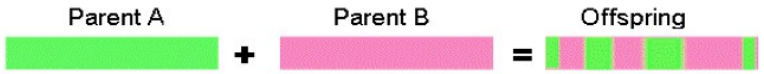
TWO POINT CROSSOVER

- Two point crossover -two crossover points are selected at random. A binary string starting from position 0 in parent A to the first crossover point is copied to the new offspring. Then a binary string starting at the first crossover point in parent B to the second crossover point in parent B is copied to the new offspring. Finally, the remaining space in the new offspring is filled with data from parent A.

11001011 + 11011111 = 11011111

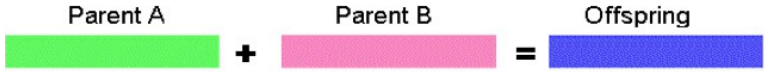
UNIFORM & ARITHMETIC CROSSOVER

- **Uniform** crossover - bits are randomly copied from the first or from the second parent.



11001011 + 11011101 = 11011111

- **Arithmetic** crossover - some arithmetic operation is performed to make a new offspring.



11001011 + 11011111 = 11001001 (AND)

BINARY MUTATION

- After a crossover is performed, mutation takes place. This prevents all solutions in population from falling into local optimums. Mutation randomly changes the new offspring.
- For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation might look as follows:

11001001 => 10001001

- The mutation depends on the encoding as well as the crossover. For example if the chromosome represented an alphabetical letter flipping a bit might not produce a valid result.

EFFECTS OF GENETIC OPERATORS

- Using selection alone will tend to fill the population with copies of the best individual from the population
- Using selection and crossover operators will tend to cause the algorithms to converge on a good but sub-optimal solution
- Using mutation alone induces a random walk through the search space.
- Using selection and mutation creates a parrallel, noise-tolerant algorithm.
- The effects of the genetic operators listed above are usually controlled by parameterising the genetic algorithm.

GENETIC ALGORITHM PARAMETERS

- **Crossover probability**

  This states how often crossover will be performed. If there is no crossover, offspring will be exact copy of their parents. If there is crossover, offspring will be made from parts of their parents chromosomes. If crossover probability is 100%, then all offspring will be produced by crossover. If it is 0%, whole new generations will be made from exact copies of chromosomes from an old population (but this does not mean that the new generation is the same!).

  Crossover is done in the hope that new chromosomes will contain the good parts of old chromosomes and that maybe these new chromosomes will be fitter. However it is a good idea to allow some part of the old population to survive in the next generation.

GENETIC ALGORITHM PARAMETERS

- **Mutation probability**

  This states how often mutation will occur. If there is no mutation, offspring is produced by crossover alone. If mutation is performed the offspring is produced by a combination of crossover and mutation. If mutation probability is 100%, whole chromosomes are changed, if it is 0%, nothing is changed.

  Mutation is used to prevent a GA from falling into local extremes. One important point to make however, is that if mutation is over used it will in effect turn the GA into a random search.

GENETIC ALGORITHM PARAMETERS

- **Population size**

  Population size determines how many chromosomes make up the population. If there are too few chromosomes, the diversity of offspring will be reduced and only a small part of the search space will be explored.

  On the other hand, if there are too many chromosomes, the GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem), it is not useful to increase population size, because it does not make solving the problem any faster.

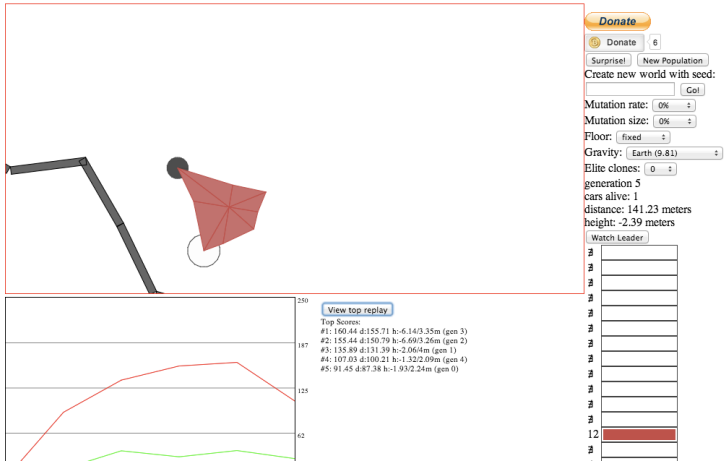
EXAMPLES

http://www.youtube.com/watch?v=xclBoPuNIiw

EXAMPLES

http://www.youtube.com/watch?v=IPQnVEnFTgY

EXAMPLES



http://rednuht.org/genetic_cars_2/