Evolutionary Computation

Lecture Notes_2

Historical perspective EC

- 1948, Turing:
  proposes "genetical or evolutionary search"
- 1962, Bremermann:
  optimization through evolution and recombination
- 1964, Rechenberg:
  introduces evolution strategies
- 1965, L. Fogel, Owens and Walsh:
  introduce evolutionary programming
- 1975, Holland:
  introduces genetic algorithms
- 1992, Koza:
  introduces genetic programming
- 1985: first international conference (ICGA)
- 1990: first international conference in Europe (PPSN)
- 1993: first scientific EC journal (MIT Press)
- 1997: launch of European EC Research Network EvoNet

EC in the early 21st Century:

- 3 major EC conferences, about 10 small related ones
- 4 scientific core EC journals
- 1000+ EC-related papers published last year(estimate)
- uncountable (meaning: many) applications
- uncountable (meaning:?) consultancy and R&D firms
- part of many university curricula

Darwinian Evolution Survival of the fittest

- All environments have finite resources (i.e., can only support a limited number of individuals)
- Life forms have basic instinct/ lifecycles geared towards reproduction
- Therefore some kind of selection is inevitable
- Those individuals that compete for the resources most effectively have increased chance of reproduction
- Note: fitness in natural evolution is a derived, secondary measure, i.e., we (humans) assign a high fitness to individuals with many offspring

Diversity drives change

Phenotypic traits:

- Behaviour / physical differences that affect response to environment
- Partly determined by inheritance, partly by factors during development
- Unique to each individual, partly as a result of random changes

If phenotypic traits:

- Lead to higher chances of reproduction
- Can be inherited

then they will tend to increase in subsequent generations, leading to new combinations of traits …

Summary

- Population consists of diverse set of individuals
- Combinations of traits that are better adapted tend to increase representation in population

Individuals are "units of selection"

- Variations occur through random changes yielding constant source of diversity, coupled with selection means that:

Population is the "unit of evolution"

- Note the absence of "guiding force"

Genetics: Natural

- The information required to build a living organism is coded in the DNA of that organism
- Genotype (DNA inside) determines phenotype
- Genes → phenotypic traits is a complex mapping
  - One gene may affect many traits (pleiotropy)
  - Many genes may affect one trait (polygeny)
- Small changes in the genotype lead to small changes in the organism (e.g., height, hair colour)

Genetics: Genes and the Genome

- Genes are encoded in strands of DNA called chromosomes
- In most cells, there are two copies of each chromosome (diploidy)
- The complete genetic material in an individual's genotype is called the Genome
- Within a species, most of the genetic material is the same

Genetics Example: Homo Sapiens

- Human DNA is organised into chromosomes
- Human body cells contain 23 pairs of chromosomes which together define the physical attributes of the individual

Genetics: Reproductive Cells

- Gametes (sperm and egg cells) contain 23 individual chromosomes rather than 23 pairs
- Cells with only one copy of each chromosome are called haploid
- Gametes are formed by a special form of cell splitting called meiosis
- During meiosis the pairs of chromosome undergo an operation called *crossing-over*

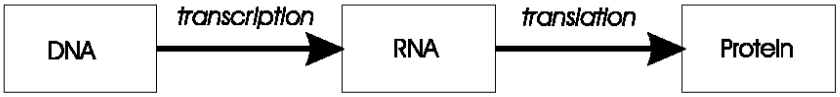
Genetics: Crossing-over during meiosis

- Chromosome pairs align and duplicate
- Inner pairs link at a *centromere* and swap parts of themselves
- Outcome is one copy of maternal/paternal chromosome plus two entirely new combinations
- After crossing-over one of each pair goes into each gamete

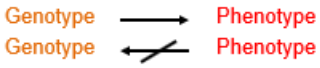
Genetics: Genetic code

- All proteins in life on earth are composed of sequences built from 20 different amino acids
- DNA is built from four nucleotides in a double helix spiral: purines A,G; pyrimidines T,C
- Triplets of these from *codons*, each of which codes for a specific amino acid
- Much redundancy:
  - purines complement pyrimidines
  - the DNA contains much rubbish
  - 4^3=64 codons code for 20 amino acids
  - genetic code = the mapping from codons to amino acids
- For all natural life on earth, the genetic code is the same!

Genetics: Transcription, translation



A central claim in molecular genetics: only one way flow



Lamarckism (saying that acquired features can be inherited) is thus wrong!

Genetics: Mutation

- inherited from either parent
- This can be
  - catastrophic: offspring in not viable (most likely)
  - neutral: new feature not influences fitness
  - advantageous: strong new feature occurs
- Redundancy in the genetic code forms a good way of error checking Occasionally some of the genetic material changes very slightly during this process (replication error)
- This means that the child might have genetic material information not

Motivation for evolutionary computing

- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solver known in nature is:
  - the (human) brain that created "the wheel, New York, wars and so on" (after Douglas Adams' Hitch-Hikers Guide)
  - the evolution mechanism that created the human brain (after Darwin's Origin of Species)
- Answer 1 → neurocomputing
- Answer 2 → evolutionary computing
- Developing, analyzing, applying problem solving methods a.k.a. algorithms is a central theme in mathematics and computer science
- Time for thorough problem analysis decreases
- Complexity of problems to be solved increases
- Consequence: ROBUST PROBLEM SOLVING technology needed

Introduction To Evolutionary Computation

This note covered the basic overview of Evolutionary Computation (EC) as a whole. we will see how EC seeks to solve problems, then we will move on to a basic overview of Evolutionary Algorithms and explain the main details. Algorithms.

Evolutionary computation is a sub-field of artificial intelligence (AI) and is used extensively in complex optimization problems and for continuous optimization. Evolutionary computation is used to solve problems that have too many variables for traditional algorithms. Computers performing evolutionary computing run such evolutionary algorithms as genetic algorithms, evolutionary programming, genetic programming and swarm intelligence models like ant colony optimization or particle swarm optimization.

The computational models using evolutionary algorithms apply evolutionary processes in order to solve complex problems. These evolutionary processes are inspired by biological evolution theory. Evolving algorithms use principles such as inheritance from previous successful generations, and natural selection where the best solutions pass their traits on to the successive generations.

How evolutionary computation works

An initial batch of possible solutions is created with the start of an evolutionary computation. The solutions once tried are refined as weaker solutions are stochastically removed and small random changes are introduced to successive generations. As the generations pass the solutions become increasingly refined. In the end the solutions produced by evolutionary computation can be tightly optimized, even though in the beginning, the approach is not understood.

Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search

provided with historical data to direct the search into the region of better performance in solution space. **They are commonly used to generate high-quality solutions for optimization problems and search problems.**

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate “survival of the fittest” among individual of consecutive generation for solving a problem.

Each generation consist of a population of individuals and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.

Foundation of Genetic Algorithms

Genetic algorithms are based on an analogy with genetic structure and behaviour of chromosomes of the population. Following is the foundation of GAs based on this analogy –

1. Individual in population compete for resources and mate
2. Those individuals who are successful (fittest) then mate to create more offspring than others
3. Genes from “fittest” parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.
4. Thus each successive generation is more suited for their environment.

Search space

The population of individuals are maintained within search space. Each individual represents a solution in search space for given problem. Each individual is coded as a finite length vector (analogous to chromosome) of components. These variable components are analogous to Genes.

Thus a chromosome (individual) is composed of several genes (variable components).

Fitness Score

A Fitness Score is given to each individual which **shows the ability of an individual to “compete”**. The individual having optimal fitness score (or near optimal) are sought.

The GAs maintains the population of n individuals (chromosome/solutions) along with their fitness scores. The individuals having better fitness scores are given more chance to reproduce than others. The individuals with better fitness scores are selected who mate and produce **better offspring** by combining chromosomes of parents. The population size is static so the room has to be created for new arrivals. So, some individuals die and get replaced by new arrivals eventually creating new

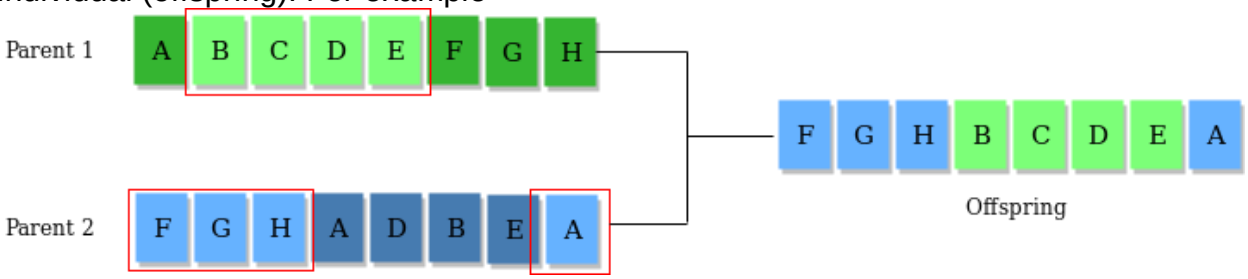
generation when all the mating opportunity of the old population is exhausted. It is hoped that over successive generations better solutions will arrive while least fit die. Each new generation has on average more “better genes” than the individual (solution) of previous generations. Thus each new generations have better **“partial solutions”** than previous generations. Once the offspring produced having no significant difference from offspring produced by previous populations, the population is converged. The algorithm is said to be converged to a set of solutions for the problem.

Operators of Genetic Algorithms

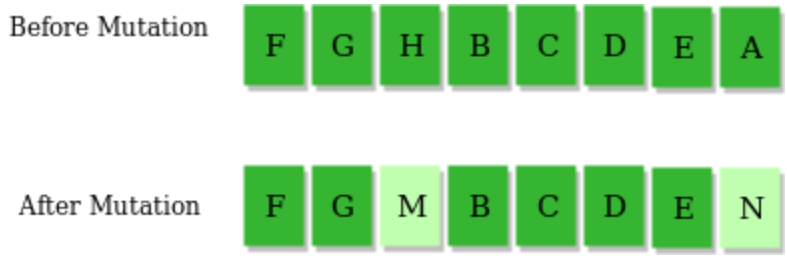
Once the initial generation is created, the algorithm evolves the generation using following operators –

1) Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations.

2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring). For example –



3) Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in the population to avoid premature convergence. For example –



The whole algorithm can be summarized as –

1) Randomly initialize populations p

2) Determine fitness of population

3) Until convergence repeat:

   a) Select parents from population

   b) Crossover and generate new population

c) Perform mutation on new population

d) Calculate fitness for new population

Example problem and solution using Genetic Algorithms

Given a target string, the goal is to produce target string starting from a random string of the same length. In the following implementation, following analogies are made –

- Characters A-Z, a-z, 0-9, and other special symbols are considered as genes
- A string generated by these characters is considered as chromosome/solution/Individual

Fitness score is the number of characters which differ from characters in target string at a particular index. So individual having lower fitness value is given more preference.