University of Babylon, College of science for women
Dept. of Computer science

Evolutionary Computing

Forth year

Dr. Salah Al-Obaidi

Lecture #8:**Popular Evolutionary Algorithm Variants**

Fall 2024

Contents

Contents **i**

5 Popular Evolutionary Algorithm Variants **46**

5.1 Genetic Algorithms 46

5.2 Evolution Strategies 48

5.3 Evolutionary Programming 49

5.4 Genetic Programming 50

6 Genetic algorithms **53**

6.1 Introduction 53

6.2 Genetic algorithm basics and some variations 54

6.3 Notion of Natural Selection 56

6.3.1 Initial Population 56

6.3.2 Fitness Function 57

6.3.3 Selection 58

6.3.4 Crossover 58

6.3.5 Mutation 59

6.3.6 Termination 60

5. Popular Evolutionary Algorithm Variants

In this lecture, we describe the most widely known evolutionary algorithm variants.

5.1 Genetic Algorithms

The genetic algorithm (GA) is the most widely known type of evolutionary algorithm. It was initially conceived by Holland in the 1960s as a means of studying adaptive behaviour, as suggested by the title of the book describing his early research: *Adaptation in Natural and Artificial Systems*. However, GAs have largely been considered as function optimisation methods. This algorithm has come to be considered as the classical genetic algorithm — commonly referred to as the 'canonical' or 'simple GA' (SGA).

SGA algorithm has a binary representation, fitness proportionate selection, a low probability of mutation, and an emphasis on genetically inspired recombination as a means of generating new candidate solutions. It is summarised in Table 5.1.

While, GAs traditionally have a fixed workflow: given a population of $\boldsymbol{\mu}$ individuals, parent selection fills an intermediary population of $\boldsymbol{\mu}$, allowing duplicates. Then, the intermediary population is shuffled to create random

Table 5.1: Sketch of the simple GA.

Representation	**Incorrect Bit-strings**
Recombination	**1-Point crossover**
Mutation	**Bit flip**
Parent selection	**Fitness proportional - implemented by Roulette Wheel**
Survival selection	**Generational**

pairs and crossover is applied to each consecutive pair with probability $\boldsymbol{p_c}$ and the children replace the parents immediately. The new intermediary population undergoes mutation individual by individual, where each of the $\boldsymbol{l}$ bits in an individual is modified by mutation with independent probability $\boldsymbol{p_m}$. The resulting intermediary population forms the next generation replacing the previous one entirely. Note that in this new generation, there might be pieces, perhaps complete individuals, from the previous one that survived crossover and mutation without being modified, but the likelihood of this is rather low (depending on the parameters $\boldsymbol{\mu}$, $\boldsymbol{p_c}$, $\boldsymbol{p_m}$).

In the early years of the field, there was significant attention paid to trying to establish suitable values for GA parameters such as the population size, crossover, and mutation probabilities. Recommendations were for mutation rates between $\mathbf{1/}\boldsymbol{l}$ and $\mathbf{1/}\boldsymbol{\mu}$, crossover probabilities around **0.6-0.8**, and population sizes in the fifties or low hundreds.

More recently it has been recognised that there are some flaws in the SGA. Factors such as elitism, and non-generational models were added to offer faster convergence if needed. As discussed in lecture #6, SUS is preferred to roulette wheel implementation, and most commonly rank-based selection is used, implemented via tournament selection for simplicity and speed.

Nevertheless, despite its simplicity, the SGA is still widely used, not just for teaching purposes, and for benchmarking new algorithms, but also for relatively straightforward problems in which binary representation is suitable.

5.2 Evolution Strategies

Evolution strategies (ES) were invented in the early 1960s by Rechenberg and Schwefel, who were working at the Technical University of Berlin on an application concerning shape optimisation. The earliest ES's were simple two-membered algorithms denoted (1+1) ES's (pronounce: one plus one ES), working in a vector space. An offspring is generated by the addition of a random number independently to each to the elements of the parent vector and accepted if fitter. An alternative scheme, denoted as (1,1) ES (pronounce: one comma one ES) always replaces the parent by the offspring, thus forgetting the previous solutions by definition. The random numbers are drawn from a Gaussian distribution with mean zero and a standard deviation $\boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ is called the mutation step size. A summary of ES is given in Table 5.2.

Table 5.2: Sketch of ES.

Representation	**Real-valued vectors**
Recombination	**Discrete or intermediary**
Mutation	**Gaussian perturbation**
Parent selection	**Uniform random**
Survival selection	**Deterministic elitist replacement by** (μ, λ) **or** $(\mu + \lambda)$

The basic recombination scheme in evolution strategies involves two

parents that create one child. To obtain $\boldsymbol{\lambda}$ offspring, recombination is performed $\boldsymbol{\lambda}$. There are two recombination variants distinguished by the manner of recombining parent alleles. Using **discrete recombination**, one of the parent alleles is randomly chosen with an equal chance for either parents. In **intermediate recombination**, the values of the parent alleles are averaged.

The selection scheme that is generally used in evolution strategies is $(\boldsymbol{\mu}, \boldsymbol{\lambda})$ selection, which is preferred over $(\boldsymbol{\mu} + \boldsymbol{\lambda})$ selection.

5.3 Evolutionary Programming

Evolutionary programming (EP) was originally developed by Fogel et al. in the 1960s to simulate evolution as a learning process with the aim of generating artificial intelligence. The classic EP systems used finite state machines as individuals.

Nowadays, EP frequently uses real-valued representations, and so has almost merged with ES. The principal differences lie perhaps in the biological inspiration: in EP each individual is seen as corresponding to a distinct species, and so there is no recombination. Furthermore, the selection mechanisms are different. In ES parents are selected stochastically, then the selection of the $\boldsymbol{\mu}$ best from the union of $\boldsymbol{\mu} + \boldsymbol{\lambda}$ offspring is deterministic. By contrast, in EP each parent generates exactly one offspring (i.e., $\boldsymbol{\lambda} = \boldsymbol{\mu}$), but these parents and offspring populations are then merged and compete in stochastic round-robin tournaments for survival. The field now adopts a very open, pragmatic approach that the choice of representation, and

hence mutation, should be driven by the problem. Table 5.3 is therefore a representative rather than a standard algorithm variant.

Table 5.3: Sketch of EP.

Representation	**Real-valued vectors**
Recombination	**None**
Mutation	**Gaussian perturbation**
Parent selection	**Deterministic (each parent creates one offspring via mutation)**
Survival selection	**Probabilistic** $(\mu + \lambda)$

5.4 Genetic Programming

Genetic programming is a relatively young member of the evolutionary algorithm family. It differs from other EA strands in its application area as well as in its particular representation (using trees as chromosomes). Although the EAs discussed so far are typically applied to optimization problems, GP could instead be positioned in machine learning. In terms of the different problem types, most other EAs are used to find some input that achieves the maximum payoff (Figure 5.1), whereas GP is used to seek models with maximum fit (Figure 5.2).

Clearly, once maximization is introduced, modeling problems can be seen as special cases of optimization. This, in fact, is the basis of using evolution for such tasks: models — represented as parse trees — are treated as individuals, and their fitness is the model quality to be maximised. The summary of GP is given in Table 5.4.

In GP, offspring are typically created by either recombination or mutation, rather than recombination followed by mutation, as is more common in other variants. This difference is illustrated in Figure 5.3, which

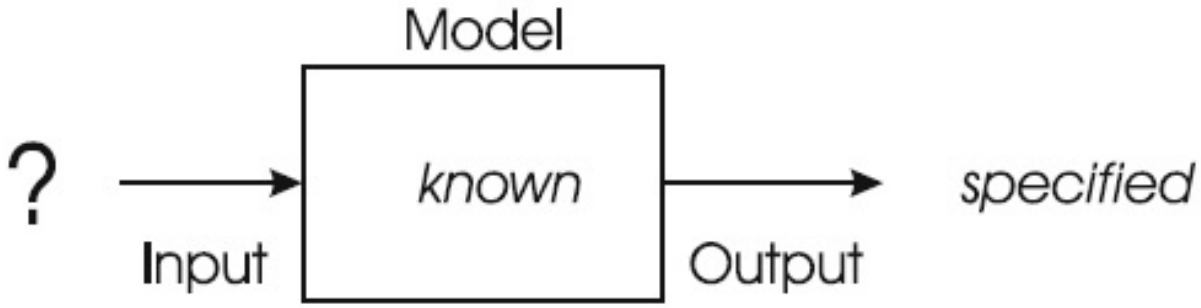

Figure 5.1: Optimisation problems. These occur frequently in engineering and design. The label on the Output reads "specified", instead of "known", because the specific value of the optimum may not be known, only defined implicitly (e.g., the lowest of all possibilities).

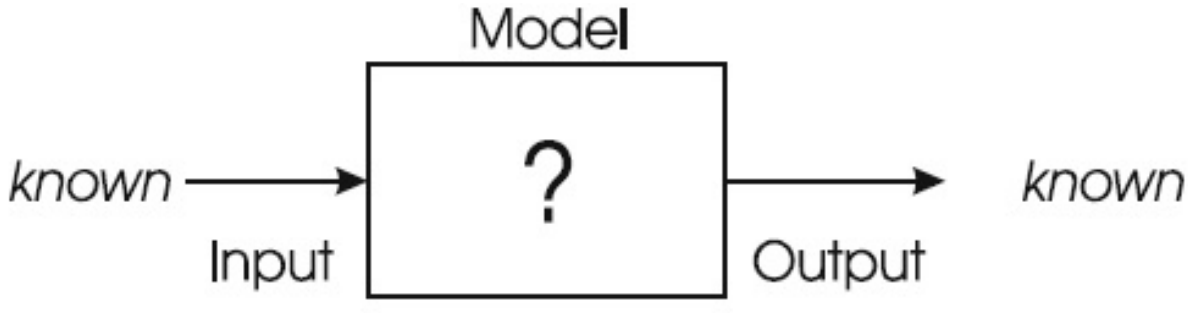

Figure 5.2: Modelling or system identification problems. These occur frequently in data mining and machine learning.

Table 5.4: Sketch of GP.

Representation	**Tree structures**
Recombination	**Exchange of subtrees**
Mutation	**Random change in trees**
Parent selection	**Fitness proportional**
Survival selection	**Generational replacement**

compares the loop to fill the next generation in a generational GA with that of GP.

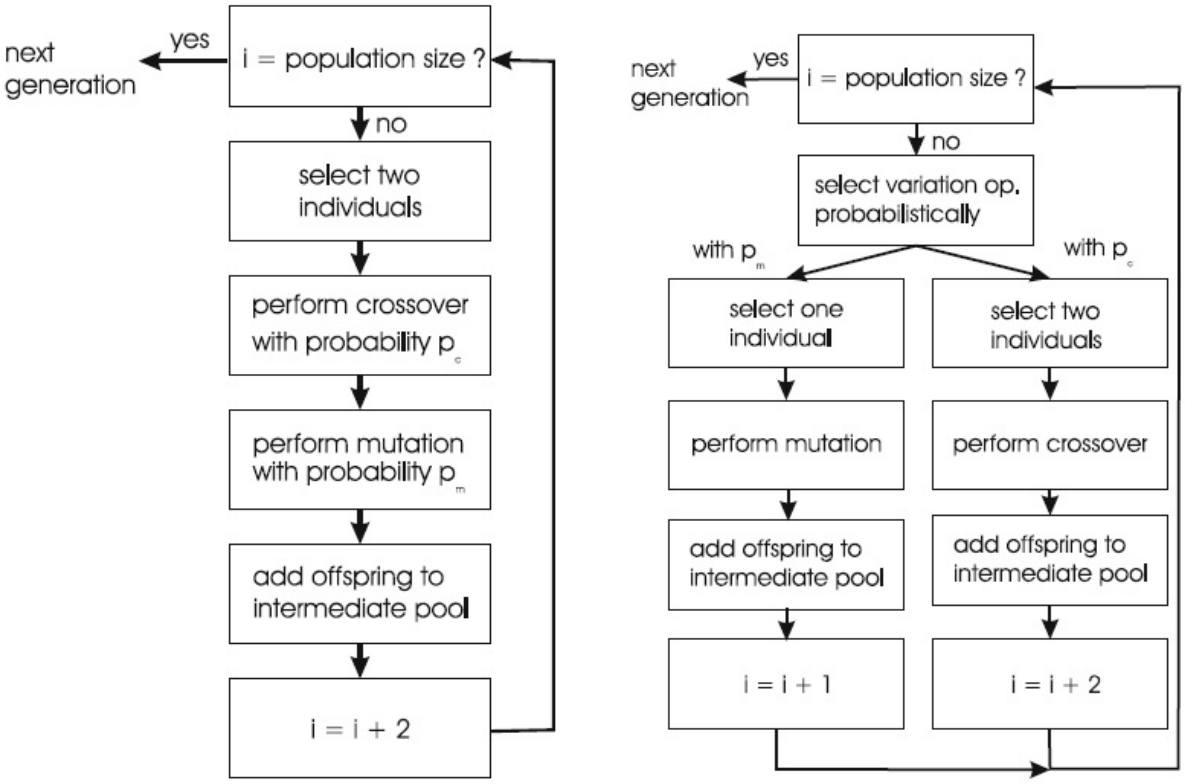

Figure 5.3: GP flowchart versus GA flowchart. The two diagrams show two options for filling the intermediary population in a generational scheme. In a conventional GA mutation and crossover are used to produce the next offspring (left). Within GP, a new individual is created by either mutation or crossover (right).

6. Genetic algorithms

6.1 Introduction

Genetic algorithms (GAs) are a class of evolutionary algorithms that was first proposed and analyzed by John Holland. There are three features that distinguish GAs, as first proposed by Holland, from other evolutionary algorithms: **(i)** the representation used—bitstrings; **(ii)** the method of selection - proportional selection; and **(iii)** the primary method of producing variations -crossover. Of these three features, however, it is the emphasis placed on crossover that makes GAs distinctive.

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection in which the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

6.2 Genetic algorithm basics and some variations

An initial population of individual structures, $P(0)$, is generated (usually randomly) and each individual is evaluated for fitness. Then, some of these individuals are selected for mating and copied to the mating buffer, $C(t)$. In Holland's original GA, individuals are chosen for probabilistic mating, assigning each individual a probability proportional to its observed performance. Thus, better individuals are given more opportunities to produce offspring (reproduction with emphasis). Next, the genetic operators (usually mutation and crossover) are applied to the individuals in the mating buffer, producing offspring $C^{'}(t)$. The rates at which mutation and crossover are applied are an implementation decision. If the rates are low enough, it is likely that some of the offspring produced will be identical to their parents. Other implementation details are how many offspring are produced by crossover (one or two), and how many individuals are selected and paired in the mating buffer. In Holland's original description, only one pair is selected for mating per cycle. The pseudocode for the genetic algorithm is shown in Figure 6.1.

After the new offspring have been created via the genetic operators the two populations of parents and children must be merged to create a new population. Since most GAs maintain a fixed-sized population $\boldsymbol{M}$, this means that a total of $\boldsymbol{M}$ individuals need to be selected from the parent and child populations to create a new population. One possibility is to

```
begin
    t = 0;
    initialize P(t);
    evaluate structures in P(t);
    while termination condition not satisfied do
    begin
        t = t + 1;
        select_repro C(t) from P(t-1);
        recombine and mutate structures in C(t)
        forming C'(t);
        evaluate structures in C'(t);
        select_replace P(t) from C'(t) and P(t-1);
    end
end
```

Figure 6.1: Modelling or system identification problems. These occur frequently in data mining and machine learning.

use all the children generated (assuming that the number is not greater than $\boldsymbol{M}$) and randomly select (without any bias) individuals from the old population to bring the new population up to size $\boldsymbol{M}$. If only one or two new offspring are produced, this in effect means randomly replacing one or two individuals in the old population with the new offspring. (This is what Holland's original proposal did.) On the other hand, if the number of offspring created is equal to $\boldsymbol{M}$, then the old parent population is completely replaced by the new population.

One popular method of performing selection is tournament selection (Goldberg and Deb 1991). A small subset of individuals is chosen at random, and then the best individual (or two) in this set is (are) selected for the

mating buffer. Tournament selection, like rank selection, is less subject to rapid takeover by good individuals.

6.3 Notion of Natural Selection

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them. Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

6.3.1 Initial Population

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem you want to solve. An individual

is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution). In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome (see Figure 6.2).

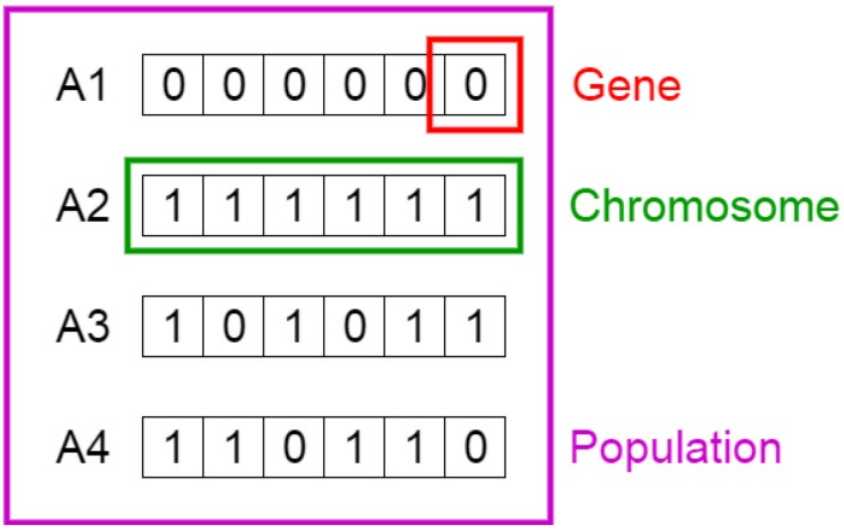

Figure 6.2: Population, Chromosomes and Genes.

6.3.2 Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

6.3.3 Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

6.3.4 Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. For example, consider the crossover point to be **3** as shown in Figure 6.3.

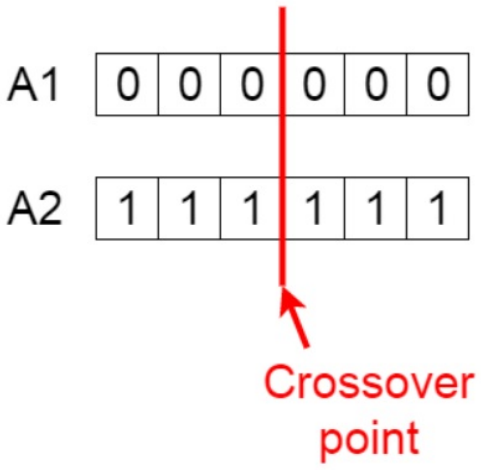

Figure 6.3: Crossover point.

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population.

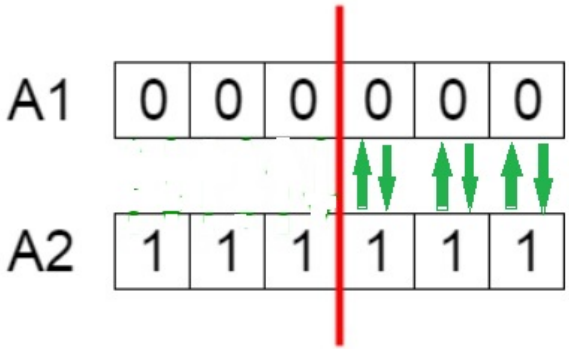

Figure 6.4: Exchanging genes among parents.

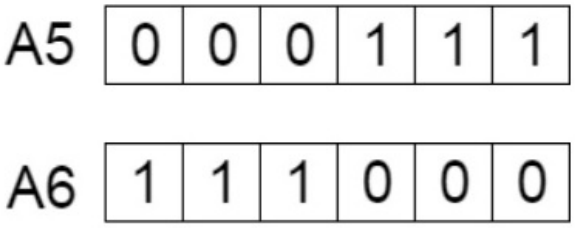

Figure 6.5: New offspring.

6.3.5 Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.

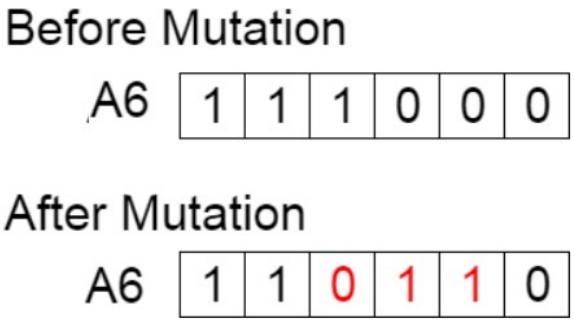

Figure 6.6: Mutation: Before and After.

Mutation occurs to maintain diversity within the population and prevent premature convergence.

6.3.6 Termination

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

***Note**, The population has a fixed size. As new generations are formed, individuals with least fitness die, providing space for new offspring. The sequence of phases is repeated to produce individuals in each new generation which are better than the previous generation.*