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

Evolutionary Computing

Dr. Salah Al-Obaidi

Lecture #3: Components of Evolutionary Algorithms

Fall 2024

Contents

2. Components of Evolutionary Algorithms

There are a number of components, procedures, or operators that must be specified in order to define a particular EA. The most important components are:

- representation (definition of individuals)
- evaluation function (or fitness function)
- population
- parent selection mechanism
- variation operators, recombination and mutation
- survivor selection mechanism (replacement)

To create a complete, runnable algorithm, it is necessary to specify each component and to define the initialization procedure. If we wish the algorithm to stop at some stage, we must also provide a termination condition.

2.1 Representation (Definition of Individuals)

The first step in defining an EA is to link the ‘real world’ to the ‘EA world’, that is, to set up a bridge between the original problem context and the problem-solving space where evolution takes place. The first step from the point of view of automated problem-solving is to decide how possible solutions should be specified and stored in a way that can be manipulated by a computer. We say that objects forming possible solutions within the original problem context are referred to as **phenotypes**, while their encoding, that is, the individuals within the EA, are called **genotypes**. This first design step is commonly called **representation**, as it amounts to specifying a mapping from the phenotypes onto a set of genotypes that are said to represent them. For instance, given an optimisation problem where the possible solutions are integers, the given set of integers would form the set of phenotypes. In this case one could decide to represent them by their binary code, so, for example, the value **18** would be seen as a phenotype, and **10010** as a genotype representing it. It is important to understand that the phenotype space can be very different from the genotype space, and that the whole evolutionary search takes place in the genotype space. A solution - a good phenotype - is obtained by decoding the best genotype after termination. Therefore it is desirable that the (optimal) solution to the problem at hand - a phenotype - is represented in the given genotype space.

The evolutionary computation literature contains many synonyms:

- On the side of the original problem context the terms **candidate**

solution, **phenotype**, and **individual** are all used to denote possible solutions. The space of all possible candidate solutions is commonly called the **phenotype space**.

- On the side of the EA, the terms **genotype**, **chromosome**, and again individual are used to denote points in the space where the evolutionary search actually takes place. This space is often termed the **genotype space**.

- There are also many synonymous terms for the elements of individuals. A placeholder is commonly called a **variable**, a **locus** (plural: loci), a **position**, or - in a biology-oriented terminology - a **gene**. An object in such a place can be called a value or an **allele**.

It should be noted that the word 'representation' is used in two slightly different ways. Sometimes it stands for the mapping from the phenotype to the genotype space. In this sense it is synonymous with **encoding**, e.g., one could mention binary representation or binary encoding of candidate solutions. The inverse mapping from genotypes to phenotypes is usually called **decoding**, and it is necessary that the representation should be invertible so that for each genotype there is at most one corresponding phenotype.

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 **1**s and **0**s). We say that we encode the genes in a chromosome. These synonyms are explained in Figure 2.1.

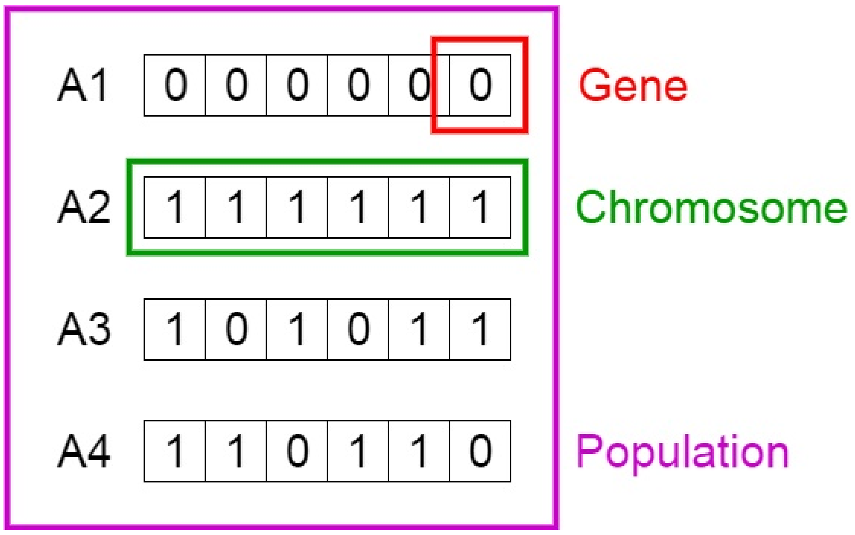

Figure 2.1: Population, Chromosomes and Genes.

The choice of the initial population size is crucial to the convergence and complexity of the algorithm. Large population sizes increase initial diversity and can maintain that diversity; however, it heavily taxes computational complexity. On the other hand, smaller sizes decrease initial diversity and diversity held throughout the generations, but it requires less computation, offering a faster alternative. The choice of size for the initial population is dependent upon the complexity of the problem, the computation cost of the fitness function, and the available waiting time.

2.2 Population

The role of the **population** is to hold (the representation of) possible solutions. A population is a multiset of genotypes. The population forms the unit of evolution. Individuals are static objects that do not change or adapt; it is the population that does. Given a representation, defining a population may be as simple as specifying how many individuals are in it, that is, setting the population size. In some sophisticated EAs, a population has an additional spatial structure, defined via a distance measure or a neighbourhood relation. This corresponds loosely to the way that real populations evolve within the context of a spatial structure given by individuals' geographical locations. In such cases, the additional structure must also be defined in order to fully specify a population.

In almost all EA applications, the population size is constant and does not change during the evolutionary search – this produces the limited resources need to create competition. The selection operators (parent selection and survivor selection) work at the population level. In general, they consider the whole current population, and choices are always made relative to what is currently present. For instance, the best individual of a given population is chosen to seed the next generation, or the worst individual of a given population is chosen to be replaced by a new one.

The **diversity** of a population is a measure of the number of different solutions present. No single measure for diversity exists. Typically people might refer to the number of different fitness values present, the number of different phenotypes present, or the number of different genotypes.

2.3 Evaluation Function (Fitness Function)

The **evaluation 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.

The role of the **evaluation function** is to represent the requirements the population should adapt to meet. It forms the basis for selection, and so it facilitates improvements. More accurately, it defines what improvement means. From the problem-solving perspective, it represents the task to be solved in the evolutionary context. Technically, it is a function or procedure that assigns a quality measure to genotypes. Typically, this function is composed from the inverse representation (to create the corresponding phenotype) followed by a quality measure in the phenotype space. To stick with the example above, if the task is to find an integer $\mathbf{x}$ that maximizes $\mathbf{x}^2$, the fitness of the genotype **10010** could be defined by decoding its corresponding phenotype ($\mathbf{10010} \rightarrow \mathbf{18}$) and then taking its square: $18^2 = \mathbf{324}$.

The evaluation function is commonly called the **fitness function** in evaluation computing. This might cause a counter intuitive terminology if the original problem requires minimization because the term fitness is usually associated with maximization. In the case of minimization, we can scale the function values such that smaller values result in larger fitness values. The selection of the fitness function is directly dependent upon what problem needs to be optimized.

Quite often, the original problem to be solved by an EA is an optimization problem. In this case, the name objective function is often used in the original problem context, and the evaluation (fitness) function can be identical to, or a simple transformation of, the given objective function.