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

Evolutionary Computing

Dr. Salah Al-Obaidi

Lecture #4: Components of Evolutionary AlgorithmsFall 2024

Contents

2. Overview of Evolutionary Algorithms

2.1 Components of Evolutionary Algorithms

2.1.4 Parent Selection Mechanism

The role of **parent selection or mate selection** is to distinguish among individuals based on their quality, and, in particular, to allow the better individuals to become parents of the next generation. An individual is a parent if it has been selected to undergo variation in order to create offspring. Together with the survivor selection mechanism, parent selection is responsible for pushing quality improvements. In EC, parent selection is typically probabilistic. Thus, *high-quality individuals have more chance of becoming parents than those with low quality.*

2.1.5 Variation Operators (Mutation and Recombination)

The role of **variation operators** is to create new individuals from old ones. In the corresponding phenotype space, this amounts to generating new candidate solutions. From the generate-and-test search perspective,

variation operators perform the generate step. Variation operators in EC are divided into two types based on their arity, distinguishing unary (mutation) and n-ary versions (recombination).

Mutation

A unary variation operator is commonly called **mutation**. It is applied to one genotype and delivers a (slightly) modified mutant, the **child or offspring**. A mutation operator is always stochastic: its output; e.g. the child, depends on the outcomes of a series of random choices. However, in general *xtmutation is supposed to cause a random, unbiased change.* Historically, mutation has played a different role in various EC dialects. Thus, for example, in genetic programming it is often not used at all, whereas in genetic algorithms it has traditionally been seen as a background operator, providing the gene pool with 'fresh blood', and in evolutionary programming it is the only variation operator, solely responsible for the generation of new individuals.

Recombination

A binary variation operator is called **recombination** or **crossover**. As the names indicate, such an operator merges information from two parent genotypes into one or two offspring genotypes. Like mutation, recombination is a stochastic operator: the choices of what parts of each parent are combined, and how this is done, depend on random drawings. Again, the role of recombination differs between EC dialects: in genetic programming it is often the only variation operator, and in genetic algorithms it is seen

as the main search operator, whereas in evolutionary programming it is never used. Recombination operators with a higher arity (using more than two parents) are mathematically possible and easy to implement, but have no biological equivalent.

The principle behind recombination is simple – *by mating two individuals with different but desirable features, we can produce an offspring that combines both of those features.* Evolutionary algorithms create a number of offspring by random recombination, and we hope that while some will have undesirable combinations of traits, and most may be no better or worse than their parents, some will have improved characteristics.

2.1.6 Survivor Selection Mechanism (Replacement)

Similar to parent selection, the role of **survivor selection** or **environmental selection** is to distinguish among individuals based on their quality. However, it is used in a different stage of the evolutionary cycle – the survivor selection mechanism is called after the creation of the offspring from the selected parents. As mentioned before, in EC the population size is almost always constant. This requires a choice to be made about which individuals will be allowed in the next generation. This decision is often based on their fitness values, favouring those with higher quality. In contrast to parent selection, which is typically stochastic, survivor selection is often deterministic. Thus, for example, two common methods are the fitness-based method of ranking the unified multiset of parents and offspring and selecting the top segment, or the age-biased approach of selecting only from the offspring. Survivor selection is also often called the **replacement**

strategy.

2.1.7 Initialisation

Initialisation is kept simple in most EA applications; the first population is seeded by randomly generated individuals. In principle, problem-specific heuristics can be used in this step, to create an initial population with higher fitness. Whether this is worth the extra computational effort, or not, very much depends on the application at hand.

2.1.8 Termination Condition

We can distinguish two cases of a suitable termination condition. If the problem has a known optimal fitness level, probably coming from a known optimum of the given objective function, then in an ideal world our stopping condition would be the discovery of a solution with this fitness. If we know that our model of the real-world problem contains necessary simplifications, or may contain noise, we may accept a solution that reaches the optimal fitness to within a given precision $\epsilon > 0$. However, EAs are stochastic and mostly there are no guarantees of reaching such an optimum, so this condition might never get satisfied, and the algorithm may never stop. Therefore, we must extend this condition with one that certainly stops the algorithm. The following options are commonly used for this purpose:

- The maximally allowed CPU time elapses.
- The total number of fitness evaluations reaches a given limit.

- The fitness improvement remains under a threshold value for a given period of time (i.e., for a number of generations or fitness evaluations).
- The population diversity drops under a given threshold.

Technically, we simply need a condition from the above list, or a similar one that is guaranteed to stop the algorithm.