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

Evolutionary Computing

Forth year

Dr. Salah Al-Obaidi

Lecture #7: Survivor Selection Fall 2024

Contents

4. Fitness and Selection

4.2 Parent Selection

4.2.6 Overselection for Large Populations

In some cases, it may be desirable to work with extremely large populations. Sometimes this could be for technical reasons – for example, there has been a lot of interest in implementing EAs using graphics cards (GPUs), which offer similar speed-up to clusters or supercomputers, but at much lower cost. However, achieving the maximum potential speed-up typically depends on having a large population on each processing node.

Regardless of the implementation details, if the potential search space is enormous, it might be a good idea to use a large population to avoid 'missing' promising regions in the initial random generation, and thereafter to maintain the diversity needed to support exploration. Often a method called over-selection is used for population sizes of **1000** and above.

In this method, the population is first ranked by fitness and then divided into two groups, the top **x%** in one and the remaining **(100 - x)%** in the other. When parents are selected, **80%** of the selection operations choose from the first group, and the other **20%** from the second.

4.3 Survivor Selection

The **survivor selection** mechanism is responsible for managing the process of reducing the working memory of the EA from a set of μ parents and λ offspring to a set of μ individuals forming the next generation. In principle, any of the mechanisms introduced for parent selection could be also used for selecting survivors. However, over the history of EC a number of special survivor selection strategies have been suggested and are widely used.

This step in the main evolutionary cycle is also called **replacement**. Replacement strategies can be categorised according to whether they discriminate on the basis of the **fitness** or the **age** of individuals.

4.3.1 Age-Based Replacement

The basis of these schemes is that the fitness of individuals is not taken into account during the selection of which individuals to replace in the population. Instead, they are designed so that each individual exists in the population for the same number of EA iterations. This does not exclude the possibly that copies of highly-fit individuals might persist in the population, but for this to happen they must be chosen at least once in the selection phase and then survive the recombination and mutation stages without being modified.

Age-based replacement is the strategy used in the simple Genetic Algorithm. Since the number of offspring produced is the same as the number of parents $(\mu = \lambda)$, each individual exists for just one cycle, and the parents are simply discarded, to be replaced by the entire set of offspring.

This is the generational model, but in fact, this replacement strategy can also be implemented in a steady-state with overlapping populations ($\boldsymbol{\lambda < \mu}$), right to the other extreme where a single offspring is created and inserted in the population in each cycle.

An alternative method of age-based replacement for steady-state Genetic Algorithm is to randomly select a parent for replacement. A straightforward mathematical argument based on the population size being fixed tells us that this probabilistic strategy has the same mean effect - that is, on average individuals live for $\boldsymbol{\mu}$ iterations. This random strategy is far more likely to lose the best member of the population than a delete-oldest strategy. For these reasons, the random replacement strategy is not recommended.

4.3.2 Fitness-Based Replacement

A wide number of strategies based on fitness have been proposed for choosing which $\boldsymbol{\mu}$ of the $\boldsymbol{\mu + \lambda}$; i.e. parents + offspring, should go forward to the next generation. Some also take age into account.

Replace worst (GENITOR): In this scheme, the worst $\boldsymbol{\lambda}$ members of the population are selected for replacement. Although this can lead to very rapid improvements in the mean population fitness, it can also lead to premature convergence as the population tends to rapidly focus on the fittest member currently present. For this reason, it is commonly used in conjunction with large populations and/or a "no duplicates" policy.

Elitism: This scheme is commonly used in conjunction with age-

based and stochastic fitness-based replacement schemes to prevent the loss of the current fittest member of the population. In essence, a trace is kept of the current fittest member, and it is always kept in the population. Thus, if it is chosen in the group to be replaced, and none of the offspring being inserted into the population has equal or better fitness, then it is kept and one of the offspring is discarded.

Round-robin tournament: This mechanism was introduced within Evolutionary Programming, where it is applied to choose $\boldsymbol{\mu}$ survivors. However, in principle, it can also be used to select $\boldsymbol{\lambda}$ parents from a given population of $\boldsymbol{\mu}$. The method works by holding pairwise tournament competitions in round-robin format, where each individual is evaluated against $\boldsymbol{q}$ others randomly chosen from the merged parent and offspring populations. For each comparison, a "**win**" is assigned if the individual is better than its opponent. After finishing all tournaments, the $\boldsymbol{\mu}$ individuals with the greatest number of wins are selected. Typically, $\boldsymbol{q = 10}$ is recommended in Evolutionary Programming.

$(\boldsymbol{\mu + \lambda})$ Selection: The name and the notation of the $(\boldsymbol{\mu + \lambda})$ selection comes from Evolution Strategies. In general, it refers to the case where the set of offspring and parents are merged and ranked according to (estimated) fitness, then the top $\boldsymbol{\mu}$ are kept to form the next generation. This strategy can be seen as a generalisation of the GENITOR method $(\boldsymbol{\mu > \lambda})$ and the round-robin tournament in Evolutionary Programming $(\boldsymbol{\mu = \lambda})$. In Evolution Strategies $\boldsymbol{\lambda > \mu}$

with a great offspring surplus.

(μ, λ)Selection: The (μ, λ) strategy used in Evolution Strategies where typically $\lambda > \mu$ children are created from a population of μ parents. This method works on a mixture of age and fitness. The age component means that all the parents are discarded, so no individual is kept for more than one generation (although of course copies of it might exist later). The fitness component comes from the fact that the λ offspring are ranked according to the fitness, and the best μ form the next generation.

In Evolution Strategies, (μ, λ) selection, is generally preferred over $(\mu + \lambda)$ selection for the following reasons:

- The (μ, λ) discards all parents and is therefore in principle able to leave (small) local optima. This may be advantageous in a multimodal search space with many local optima.
- If the fitness function is not fixed, but changes in time, the $(\mu + \lambda)$ selection preserves outdated solutions, so it is not able to follow the moving optimum well.
- $(\mu + \lambda)$ selection hinders the self-adaptation mechanism used to adapt strategy parameters.