4 Brief introduction to genetic algorithms

Among the four main classes of evolution-like and population-oriented methods of evolutionary computations [48, 60, 62, 138, 139, 172, 173, 236], that is, genetic algorithms, evolution strategies, evolutionary programming, and genetic programming, the first class plays a particularly important role. Genetic algorithms are a popular and widely used global-search paradigm based on principles imitating mechanisms of genetics, natural selection, evolution and heredity, including the evolutionary principle of survival of the fittest (to environment) individuals and extinction of the worst adapted individuals. The underlying principles of genetic algorithms were first formulated by Holland [138]. The mathematical framework was developed in the 1960s and was presented in his pioneering book [139]. An essential feature of the genetic-algorithm-based global searching of the solution domain is preserving the best possible balance between the two opposite requirements, that is, the use of the already-found best solutions and a possibly wide search of the solution domain. Genetic algorithms offer a compromise methodology, which eliminates many shortcomings of the two extreme approaches: traditional optimization techniques and random search methods. The first rely on a single-point search (that is, a migration of a single point across the search space) and are most likely to get trapped in local extrema, which inevitably are present in many practical optimization problems. The second are strategies where in fact the whole solution space is searched, but no consideration is given to those regions of the space, which offer better solution.

In Chapter 1 a general introduction to genetic algorithms was given. The objective of this chapter is to briefly present them as an important supportive tool in the parameter (and possibly structure) learning of the computational intelligent systems presented in this book. First, major components of genetic algorithms will be briefly presented and then, a theoretical introduction to genetic computing will be outlined (on the basis of [147,183,196]).
4.1 Basic components of genetic algorithms

Genetic algorithms operate on generic structures called chromosomes. A chromosome is a binary string (a gene string), which represents a point in the solution (search) space. Each chromosome is associated with a "fitness" value that evaluates the performance of a possible solution the chromosome represents. Instead of a single point, genetic algorithms usually keep a set of points as a population, which is then evolved repeatedly toward a better overall fitness value. Genetic algorithms solve the problem of finding good chromosomes by manipulating the material in the chromosomes blindly without any knowledge about the type of problems they are solving. The only information they are given is an evaluation of each chromosome they produce by means of a fitness function. This evaluation is used to bias the selection of chromosomes so that those with the best evaluations tend to reproduce more often than those with bad evaluations. Genetic algorithms, using simple manipulations of chromosomes such as simple encodings and reproduction mechanisms, can display complicated behaviour and solve some extremely difficult problems without knowledge of the decoded world.

In each generation, the genetic algorithm constructs a new population of chromosomes using genetic operations such as crossover and mutation. The chromosomes with higher fitness values are more likely to survive and to participate in mating (crossover) operations. After a number of generations, the population contains chromosomes with better fitness values; this is analogous to Darwinian models of evolution by random mutation and natural selection. Genetic algorithms as well as other evolutionary computation methodologies are sometimes referred to as methods of population-based or population-oriented optimization that improve performance by upgrading the entire population rather than their individual members.

Genetic algorithms have the following main features:

1. They do not directly process the parameters of a given problem but, rather, their encoded representation.

2. They perform the searching of the solution space working not with a single point but with a population of points; they search many peaks in the solution space in parallel. By employing genetic operators, they exchange information between peaks, thus lessening the possibility of ending at a local extremum and missing the global extremum.

3. They need to evaluate only the fitness function to guide their search, and there is no requirement for derivatives of the fitness function or