Chapter 3

Evolutionary Computation

"survival of the fittest" -
The Origin of Species
by Charles Darwin, 2001

This chapter gives an overview of evolutionary computation (EC) [2]. The different recombination operators, selection methods and mutation are explained. The chapter also summarises the different EC paradigms. Section 3.1 provides a summary of a general evolutionary algorithm (EA), while succeeding sections provide a more detailed discussion of aspects of evolutionary computation. In these sections the focus will be on genetic algorithms, since this EC paradigm is used in the GAIS classifier to evolve a set of ALCs.

Evolutionary computation mimics natural evolution in biological organisms. Evolutionary algorithms (EA) are stochastic search algorithms where the search for an optimal solution is guided by the principle of survival of the *fittest* and mathematical models of genetic and behavioral inheritance. A population of candidate solutions (or individuals) is evolved for a number of *generations* through application of a number of operators such as *crossover*, *mutation* and *selection* until an optimal, or best individual is found as solution to the optimisation problem. The four most common specific paradigms of EC are [72]: Genetic Algorithms (GA), Genetic Programming (GP), Evolutionary Programming (EP) and Evolutionary Strategies (ES). These paradigms use the same general evolutionary algorithm.

3.1 A General Evolutionary Algorithm

The following pseudo-code algorithm summarises a general evolutionary algorithm. Although all operator types are included in the algorithm, different EC paradigms use different operators and different representations of the chromosome. The differences between the most popular EC paradigms are discussed in section 3.2.

General EA:

1. set the generation counter, $g=0$
2. initialise a population of K chromosomes
3. while no convergence
   (a) evaluate the fitness of each chromosome in the population
   (b) perform crossover
      i. select parents
      ii. produce offspring from the selected parents
   (c) perform mutation
      i. select a candidate chromosome
      ii. mutate selected candidate
   (d) select the survivors to form the next generation
   (e) evolve the next generation ($g=g+1$)

Several aspects of the general algorithm above need to be explained in more detail, for example the initialisation of the population, the fitness of a chromosome, the crossover operator and mutation on chromosomes, the selection of survivors for the next generation and the convergence in the population. These aspects are discussed in sections 3.3 to 3.8. First, the differences between EC paradigms are discussed in section 3.2.

3.2 EC Paradigms

This section outlines the main differences between the most popular EC paradigms. Other paradigms include differential evolution [66], cultural evolution [42] and co-evolution [25].

3.2.1 Genetic Algorithms (GA)

Genetic algorithms were first introduced by Alex Fraser [32], while Holland laid down the basic principles in 1975 [41]. The genetic algorithm mimics genetic evolution [4]. Chromosomes are in genotype-space and are mostly bit-strings. Continuous-valued variables are usually coded into a binary representation. A drawback of bit-string chromosomes is the occurrence of Hamming cliffs [26], which is addressed by using Hamming bit-string representations. The genetic algorithm uses recombination operators, mutation and selection methods to evolve to the best solution [26, 72].

3.2.2 Genetic Programming (GP)

Genetic programming also models genetic evolution but the chromosomes represent executable programs in a tree structure. Genetic programming was developed by Koza [52]. The goal of the genetic programming algorithm is to evolve the best executable program in a problem domain that performs (or executes) best. Therefore the fitness function measures how well a chromosome executes in the problem domain. Crossover is implemented by swapping subtrees between selected parents to produce offspring. Mutation can be implemented as randomly changing a node, expanding (growing) the tree to a larger depth, truncating the tree or randomly replace a terminal node [26, 72].

3.2.3 Evolutionary Programming (EP)

Different from genetic algorithms, evolutionary programming models behavioral evolution and chromosomes represent solutions in phenotype-space. EP was developed by Fogel [27]. EP does not use crossover, since each individual in the population generates one offspring through the mutation operator. The offspring compete against the chromosomes of the previous generation for survival to the next generation. The EP uses *elitism* as replacement scheme [26, 72].

3.2.4 Evolutionary Strategies (ES)

Evolutionary strategies model the evolution of evolution [67, 73]. That is, ES is concerned with optimising the evolutionary process itself [26, 68]. Chromosomes present solutions in both genotype- and phenotype-space, but the fitness of the chromosomes is the behavior of the individual in phenotype-space. The chromosome consists of genetic material and strategy parameters. The ES algorithm evolves the chromosome's strategy parameters, and the strategy parameters is

used to evolve the genetic material of the chromosome. The ES algorithm also uses mutation as operator and a mutated chromosome is only accepted if mutation has improved the fitness of the individual [26, 72]. The ES uses different crossover, mutation and selection techniques [26].

3.3 The Chromosome

An evolutionary algorithm makes use of a population of chromosomes, or individuals. Each chromosome represents a potential solution to the problem that needs to be optimised. A chromosome consists of the set of parameters to the problem or function that needs to be optimised. The parameters are known as *genes* and represent values in the same space as the function being optimised (referred to as *phenotype-space*). Each gene consists of *alleles*. Alleles are specific values from the domain of the corresponding parameter assigned to the gene.

There are different chromosome representation schemes. Some of these are binary string representations where the binary values are discretised real numbers or boolean values, trees that represent programs, or the chromosome can represent real-valued variables. The chromosomes converge in the limit and the best chromosome is chosen as a solution or an approximate solution to the problem. Parameter-values are usually mapped to an intermediate space, referred to as *genotype-space*. This mapping from phenotype-space to genotype-space is known as *coding* and the inverse mapping is known as *decoding* [72].

When the problem that needs to be optimised is a mathematical function, then the chromosome will represent the real-valued variables of that function. For a more complex problem like the optimisation of the execution of a program, the chromosomes will represent different programs using a tree-structure. The coding mappings can influence the global behavior of the EA algorithm. It is important to include all necessary parameters in the chromosome representation to prevent evolution to a less optimal solution. Operations on the chromosome produces offspring to widen the search space of solutions. The chromosome's genotypes are evaluated on the corresponding phenotype to determine the chromosome's fitness. At each generation, the chromosome's fitness is calculated and only the fittest chromosomes survive to the next generation. The next section discusses the fitness function, followed by methods of offspring creation.

3.4 Calculating the Fitness

The fitness function maps the chromosome's representation into a scalar value. The scalar value of the chromosome indicates how close a chromosome is to the optimal solution. The fitness function therefore provides a quantification of the quality of the chromosome. It is the fitness of a chromosome that determines whether the chromosome will be selected to produce offspring and quantifies its chances for survival among the other chromosomes in the population to the next generation. The probability to mutate a chromosome is usually a function of the chromosome's fitness. Chromosomes with high scalar fitness values should preferably not be mutated.

The fitness function is problem-specific. The function can either be a unimodal function or a multi-modal function. A unimodal function has a single optimal solution where a multi-modal function has multiple optima. The standard optimisation methods only find a single solution and for local optimisation algorithms this solution is either a local minimum or local maximum, thus not necessarily the best solution [9]. Global optimisation algorithms find the best solution. A technique known as *niching* has been developed to find multiple solutions in multi-modal functions. GAs have been successfully applied with *niching*, with the effect that individuals converge to different solutions, or *niches* [43, 53]. The best individual per *niche* is one of the solutions to the problem. Niching can be done in two ways: Parallel and sequential niching. Parallel niching concurrently finds niches in the search space through strategies that identify and refine potentially good solutions over time [35, 43, 53, 57]. Sequential niching develops niches over time, in sequence. After each discovered niche, individuals are repelled from the area around the niche to focus on unexplored areas in the search space [3].

Some problems have several objectives which are represented as sub-objectives in a multi-objective function. In most cases the sub-objective functions are in conflict, i.e. the reduction of one objective results in the increase of another objective. Multi-objective optimisation techniques find solutions with a good trade-off between the conflicting sub-objectives. Multi-modal functions and multi-objective problems have the same goal of finding multiple solutions for the optimisation problem. Multi-objective optimisation techniques find multiple solutions for a number of sub-objectives as niching has the objective to find multiple solutions for a single objective for a multi-modal problem. It is of utmost importance to include all necessary objectives in the fitness function, to prevent evolving a less optimal solution for the problem. For more information on multi-objective optimisation, the reader is referred to [13].

Before the fitness of a chromosome can be calculated, the chromosome's representation first needs to be decoded to phenotype-space if the domain of the fitness function is in phenotype-space. If the fitness function is too complex to evaluate, an approximate function evaluation can be used, which approximately gives the same value as the "true" fitness function in less time [12]. Penalty functions have also been considered in the evaluation of fitness, penalising a chromosome for an invalid or "bad" solution to the problem or a chromosome that violates a certain set of restrictions [34, 69].

3.5 Reproduction

Reproduction is the process of producing new offspring from selected individuals through crossover or mutation. The reproduction step consists of a selection step where parents are selected and a crossover step where genetic material of individuals are exchanged to form offspring. These operators are discussed in more detail in this section.

3.5.1 Selection

This section describes the most popular selection methods that can be used to select chromosomes as parents for crossover, individual chromosomes for mutation or the chromosomes that survive to the next generation. Usually, chromosomes with high fitness are selected for crossover to converge faster to a best solution. Highly fit chromosomes should not be selected for mutation to prevent the danger of diverging from "good" solutions in the search space. Therefore chromosomes with low fitness are usually selected for mutation.

All selection methods are based on the fitness of the chromosomes [10]. Certain selection methods allow selection of a chromosome more than once, resulting in clones of the selected chromosome. An advantage of clones is that more chromosomes with high fitness form part of the population, while a disadvantage is the probability of less diversity in the search space. When selecting chromosomes to survive to the next generation, either all offspring are selected to replace some or all parents from the previous generation, or chromosomes from both the offspring and the parents from the previous generation are considered for selection. Offspring can only replace parents if the offspring has a higher fitness than the parents. Goldberg and Deb [36] compared many of these selection methods. The most popular selection techniques are listed below:

Random Selection: All the individuals in the population have an equal chance to be selected with no reference to their individual fitness.

Proportional Selection: The probability of an individual to be selected is based on the individual's fitness proportional to the summed fitness of all the individuals in the population. A drawback of this selection method is that an individual may dominate the production of offspring which results in a limited diversity among individuals in the new population. This drawback can be overcome by limiting the number of offspring the selected individual may produce.

Rank-Based Selection: The individuals in the population are sorted according to their fitness. The rank ordering of the individuals in the sorted set determines the probability to select an individual. Selection is thus not based on the magnitude of an individual's fitness. As such, ranking has the advantage that a highly fit individual will not dominate the selection process. An alternative approach is to assign survival probabilities to the individuals in the sorted set using an exponential function with the rank as parameter to the exponential function. This results in a higher selection intensity with the disadvantage of less diversity which could lead to suboptimal solutions.

Tournament Selection: A random sample of i individuals is selected from the population to take part in a tournament of selecting the individual with the best fitness. The sample may be taken with or without replacement. It is therefore possible that an individual can combine with itself to produce offspring or a parent can be selected more than once to produce offspring. An advantage of tournament selection is that the individuals with worst fitness will not be selected and as long as sampling is done with replacement, the best individual will not dominate the reproduction process.

Elitism: This selection method is used to select a set of the best individuals from the previous population that will survive to the next generation. The number of individuals in the selected set is known as the *generation gap*, ς. These chromosomes are not mutated. If $\varsigma = 0$, then the new generation consists entirely out of new individuals. If $\varsigma > 0$, then ς individuals survive to the next generation. These ς selected individuals are either the ς best individuals to ensure that the maximum fitness in the population does not decrease or ς individuals selected with one of the selection methods explained above.

3.5.2 Crossover

The crossover operator exchanges genetic material between two or more selected parents to produce offspring. The main idea of the crossover operator is to recombine genetic material between fit chromosomes from previous generations with a certain probability, p_c, to produce even fitter offspring. The crossover process is described by the following general algorithm:

1. generate $\xi \sim U(0,1)$
2. select parents A and B
3. if $\xi > p_c$ then crossover is not performed and A and B are returned, otherwise goto step 4
4. exchange genetic material between A and B according to one of the crossover operators (described below)
5. return the produced offspring

The following section explains how the crossover operator is applied to chromosomes with different representations.

3.5.2.1 Continuous-valued Chromosomes

Arithmetic crossover can be used if genes are continuous-valued [5]. A number of arithmetic operators have been developed, for example:

- Average - take the arithmetic average of the genes in parents A and B as the new value for the gene in the offspring O. That is, $O_i = \frac{A_i + B_i}{2.0}$, where i is the index to the i-th gene in the chromosome.
- Geometric mean - take the square-root of the product of the two gene-values in parents A and B as the new value for the gene in the offspring O. That is, $O_i = \sqrt{A_i * B_i}$, where i is the index to the i-th gene in the chromosome.
- Extension - take the difference between the two gene-values in parents A and B, add the difference to the higher of A or B, or subtract the difference from the lower of A or B. That is,

$$O_i = Max(A_i, B_i) + y$$

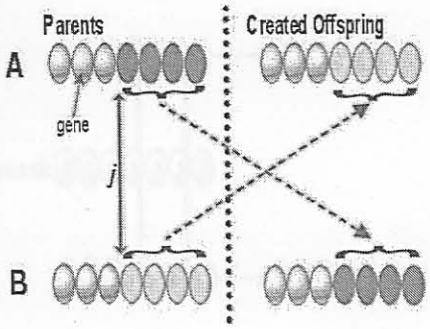

Figure 3.1: One-point crossover

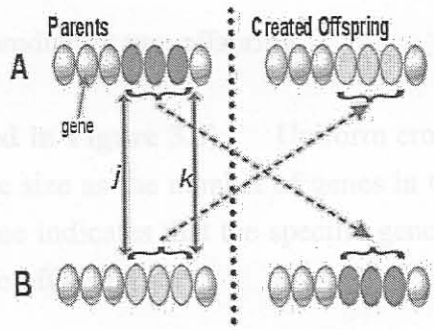

Figure 3.2: Two-point crossover

or

$$O_i = Min(A_i, B_i) - y$$

where $y = |A_i - B_i|$.

3.5.2.2 Nominal-valued Chromosomes

Nominal-valued chromosomes have genes which assume values from a finite set of values. Binary-valued genes is a special case, where the set of values contains only two values. Popular crossover operators include [26]:

One-point crossover (illustrated in Figure 3.1): A random position, j, is selected in both parents A and B. The tail of parent A is swapped with the tail of parent B, producing two offspring.

Two-point crossover (illustrated in Figure 3.2): Two random positions, j and k, are selected in both parents A and B. The selected segment between j and k in parent A is swapped with the

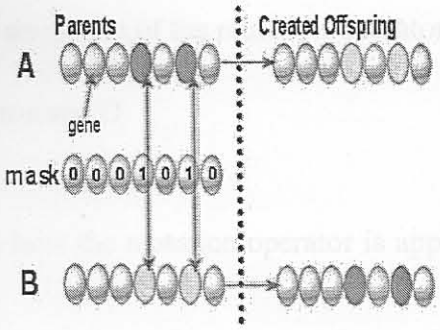

Figure 3.3: Uniform crossover

selected segment in parent B, producing two offspring.

Uniform crossover (illustrated in Figure 3.3): Uniform crossover uses a randomly generated bit mask that has the same size as the number of genes in the chromosomes. The gene in the mask that has a value of one indicates that the specific genes have to be swapped between parents A and B, producing two offspring.

Crossover prevents the evolutionary process to randomly search for the optimal solution. A high p_c can result in premature convergence to suboptimal solutions. If p_c is too low, the evolutionary process may tend to have low diversity in the search space. This may restrict the possibility to find an optimal solution in the search space.

3.6 Mutation

The main objective of mutation is to introduce diversity into a population of chromosomes with a certain probability p_m. The mutation operator randomly changes the genetic representation of the selected chromosome to ensure diversity and to cover larger parts of the search space. Mutation is only used in certain EC paradigms, where it is considered as a background operator. In these EC paradigms (excluding EP), mutation operates on offspring that have been produced by the crossover operator. The following algorithm summarises mutation:

1. select a chromosome D to mutate
2. for each gene in D generate $\xi \sim U(0,1)$
   (a) if $\xi > p_m$ then do not mutate the gene, otherwise goto 2.(b)

(b) Mutate the gene by using one of the mutation operators (described below)

3. return the mutated chromosome D

The following section explains how the mutation operator is applied to chromosomes with different representations.

3.6.1 Continuous-valued Chromosomes

Arithmetic mutation can be used if the genes of the chromosome are continuous-valued [5]. Arithmetic mutation has been implemented in the following ways:

- Random replacement, which replaces the value of a gene with a new random value that is valid in the domain of the variable.
- Creep mutation, where a small random value is added or subtracted from the value of a gene.
- Geometric creep mutation, where the value of a gene is multiplied with a value close to 1.

The random values can be sampled from any of a number of probability distributions, e.g. uniform (within a specified range), exponential, Gaussian, normal, binomial, etc. Usually, a Gaussian distribution is used with a zero mean. To obtain a small value a small variance is used. The variance of the distribution is usually a function of the fitness of the individual that needs to be mutated. An individual with a high fitness will be mutated less than an individual with a lower fitness, which will be mutated more.

3.6.2 Nominal-valued Chromosomes

Different mutation operators for nominal-valued genes are [26]:

Random Mutate (illustrated in Figure 3.4): Random genes in the chromosome are selected and each gene's value is replaced with its complement or a new random value (with probability p_m).

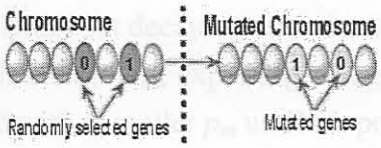

Figure 3.4: Random Mutation

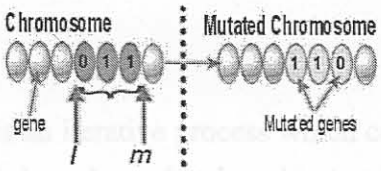

Figure 3.5: In-order Mutation

In-order Mutate (illustrated in Figure 3.5): Two random positions, l and m, are selected in the chromosome and only genes between these positions are considered for replacement using random mutation.

A high p_m causes large diversity in the search space, since new genetic material is more rapidly introduced. A large mutation rate may cause good genetic material to be lost. A high mutation rate is however beneficial in situations where more rapid coverage of the search space is needed. Usually, one starts with a large initial mutation rate, decreasing it over time. A high mutation rate basically results in a random search.

3.7 The Initial Population

The initial population is generated by randomly selecting valid values for the genes from the domain of the corresponding variables. Random selection ensures that the initial population of chromosomes has a good uniform coverage of the search space. Domain knowledge of the search space can be used to initialise the values of the genes, using heuristics to bias the initial population to potentially good solutions. The drawback of using domain knowledge is that the potential good areas in the search space are missed. A better solution may exist in the missed areas. The size of the initial population influences the convergence speed of the algorithm. With a large population size, the search space is well covered. The larger diversity may result in less generations with longer time per generation to converge to the best solution. The consequence of a small population size is less diversity in the search space. The small population may take more generations to converge but with less time per generation. To explore a larger search space with

a small population size, an initial large, but decaying mutation rate can be used. The individuals will then initially be mutated with a large p_m, exploring a larger search space. As generations pass by the individuals are mutated with a smaller p_m until the population converges to an optimal solution.

3.8 Convergence

An evolutionary algorithm (EA) is an iterative process which continues until a convergence criterion is satisfied. Several methods have been developed to test if an EA should terminate. Some of the most frequently used criteria are summarised below:

- The EA terminates when a maximum number of generations is exceeded. When the maximum number of generations is too small, the EA might not have evolved an optimal solution yet, i.e. no convergence in the population. When the maximum number of generations is too high, the population in the EA might have already converged to an optimal solution before reaching the maximum number of generations, thus wasting computational effort.
- The desired, or an acceptable best chromosome evolves. When the optimum solution to a problem is known, it can be used as a measure to determine when a suitable or satisfying solution has evolved.
- The average fitness of the population and the variance of the population's fitness do not change over a certain number of generations. This indicates that the fitness of the population has stabilised.

3.9 Conclusion

This chapter gave an overview on evolutionary computation (EC). The chapter presented the different recombination operators, selection methods and also gave a summary of the different EC paradigms. EC methods require only the values of the function that needs to be optimised. This fact makes EC methods stochastic, which can potentially find the global optimum of the objective function. EC methods search in parallel for a global solution to the problem. A disadvantage of EC methods is the computational cost, since a large number of function evaluations must be performed to find a satisfying result. The choice of evolutionary operators, chromosome representation and fitness function have a critical impact on the performance of EC methods.

Since EC methods are stochastic in nature there is no guarantee of convergence to the optimum solution. An alternative method to solve complex problems is the artificial immune system.

The next chapter gives background on existing artificial immune system models and their application to non-biological environments. An artificial immune system is a dynamic distributed system that is capable to learn and recognise patterns. The AIS is robust, scalable and tolerant to noise.