CHAPTER 1

Evolutionary Algorithms: Basic Concepts and Applications in Biometrics

Carlos A. Coello Coello
CINVESTAV-IPN
Evolutionary Computation Group
Dpto. de Ing. Elect./Secc. Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07300, MEXICO
E-mail: ccoello@cs.cinvestav.mx

In this chapter, we provide a short introduction to evolutionary algorithms, including their basic concepts and some of their representative applications reported in the specialized literature. Then, we describe some case studies involving successful applications of evolutionary algorithms in biometrics. The final part of the chapter presents some possible research directions regarding the use of evolutionary algorithms in biometrics.

1. Introduction

Nature has inspired man since ancient times. Although sometimes this inspiration was simply imitated, many of the real breakthroughs of humankind occurred when nature's principles were understood rather than copied. Many of such examples are well-documented in engineering $^{French(1994)}$ (e.g., dams, tunnels, and airplanes).

Thus, it is not by any means surprising to find that the evolution of species has served as inspiration to propose search and optimization techniques. After all, nature has evolved (sometimes rather complex) solutions to a wide variety of difficult problems over millions of years $^{Dawkins(1990)}$. Early suggestions of the connections between evolution and optimization can be traced as long back as the 1930s $^{Cannon(1932)}$. However, it was until the 1960s when the three main techniques based on the evolution of species were developed $^{Fogel(1998)}$. These approaches (genetic algorithms, evolutionary programming and evolution strategies), which are now collectively denominated "evolutionary algorithms", have been found to be very effective for single-objective optimization and are now widely used in a great variety of disciplines [Goldberg(1989),Schwefel(1981),Fogel(1995),Holland(1975),Eiben(2003)].

Biometrics is a discipline that measures and statistically analyses biological data. Recently, and in the context of information technology, the term has been adopted to refer to the technologies for measuring and analyzing human body characteristics such as fingerprints, eye retinas and irises, voice patterns, facial patterns and hand measurements, especially for authentication purposes [Zhang(2000)]. Biometric applications involve several complex problems. For example, many current biometric applications are closely related to pattern recognition and image analysis [Soldeketal.(1997)]. The complexity of these problems (which tend to be approached using statistical techniques) makes attractive the use of heuristics such as evolutionary algorithms, which have been found to be very powerful in a wide variety of optimization and classification tasks [Fogel(1995),Goldberg(1989),Eiben(2003),Bäcketal.(1997)].

The remainder of this chapter is organized as follows. Section 2 provides some basic concepts related to evolutionary algorithms. Section 3 attempts to summarize the material from the previous section, by providing a more general framework for studying evolutionary algorithms. This includes a discussion of some of the main advantages offered by evolutionary algorithms. Section 4 discusses a few representative case studies of applications of evolutionary algorithms in biometrics. After that, we provide some possible future research directions in Section 5 and our conclusions in Section 6.

2. Basic Notions of Evolutionary Algorithms

The famous naturalist Charles Darwin defined *Natural Selection* or *Survival of the Fittest* as the *preservation of favorable individual differences and variations, and the destruction of those that are injurious* [Darwin(1882)]. In nature, individuals have to adapt to their environment in order to survive in a process called *evolution*, in which those features that make an individual more suited to compete are preserved when it reproduces, and those features that make it weaker are eliminated. Such features are controlled by units called *genes* which form sets called *chromosomes.* Over subsequent generations not only the fittest individuals survive, but also their fittest genes which are transmitted to their descendants during the sexual recombination process which is called *crossover.*

Early analogies between the mechanism of natural selection and a learning (or optimization) process led to the development of the so-called "evolutionary algorithms" (EAs) $^{Bäck(1996)}$, in which the main goal is to simulate the evolutionary process in a computer. There are three main paradigms within evolutionary algorithms, whose motivations and origins were totally independent from each other: evolution strategies $^{Schwefel(1981)}$, evolutionary programming $^{Fogel(1999)}$, and genetic algorithms $^{Holland(1975)}$. Additionally, some authors consider genetic programming $^{Koza(1992)}$ as another paradigm, although genetic programming can also be seen as a special type of genetic algorithm. Each of these four types of evolutionary algorithm will be discussed next in more detail.

2.1. *Evolution Strategies*

When working towards his PhD degree in engineering at the Technical University of Berlin, Ingo Rechenberg (see Fig. 1) came across some optimization problems in hydrodynamics that could not be solved using traditional mathematical programming techniques $^{Rao(1996)}$.

Fig. 1. Ingo Rechenberg. He developed an optimization algorithm which consisted of applying a set of random changes to a reference solution. The approach was later called "evolution strategy".

This led him to the development of a very simple optimization algorithm which consisted of applying a set of random changes to a reference solution. The approach was later called "evolution strategy" and it was formally introduced in 1964 $^{Fogel(1998)}$. The original evolution strategy was called $(1+1)$-ES, because it consisted of a single parent that was mutated (i.e., subject to a random change) to produce an offspring. Then, the parent was compared to its offspring and the best from them was selected to become parent for the following iteration (or generation).

In the original (1+1)-EE, a new individual was produced using:

$$\bar{x}^{t+1} = \bar{x}^t + N(0, \vec{\sigma}),$$

where t refers to the current *generation* (or iteration) and $N(0, \vec{\sigma})$ is a vector of independent Gaussian numbers with median zero and standard deviation $\vec{\sigma}$. It is important to emphasize that an "individual" in an evolution strategy contains the set of decision variables of the problem. No encoding is used in this case. So, if the decision variables are real numbers, such real numbers are directly put together as a single vector for each individual.

Rechenberg $^{Rechenberg(1973)}$ stated a rule for adjusting the standard deviation in a deterministic way such that the evolution strategy could converge to the global optimum. This is now known as the "1/5 success rule", and it consists of the following:

$$\sigma(t) = \begin{cases} \sigma(t-n)/c \text{ if } p_s > 1/5 \\ \sigma(t-n) \cdot c \text{ if } p_s < 1/5 \\ \sigma(t-n) \quad \text{if } p_s = 1/5 \end{cases}$$

where n is the number of decision variables, t is the current generation, p_s is the relative frequency of successful mutations (i.e., those mutations in which the offspring replaced its parent because it had a better fitness) measured over a certain period of time (e.g., at every $10 \times n$ individuals) and $c = 0.817$ (this value was theoretically derived by Hans-Paul Schwefel $^{Schwefel(1981)}$ —see Fig. 2—). $\sigma(t)$ is adjusted at every n mutations.

Over the years, several other variations of the original evolution strategy were proposed, after the concept of population (i.e., a set of solutions) was introduced $^{Bäck(1996)}$. The most recent versions of the evolution strategy are the $(\mu + \lambda)$-ES and the (μ, λ)-ES. In both cases, μ parents are mutated to produce λ offspring. However, in the first case (+ selection), the μ best individuals are selected from the union of parents and offspring. In the second case (, selection), the best individuals are selected only from the offspring produced.

In modern evolution strategies, not only the decision variables of the problem are evolved, but also the parameters of the algorithm itself (i.e., the standard deviations). This is called "self-adaptation"

Fig. 2. Hans-Paul Schwefel. He theoretically derived the “1/5 success rule”, which is used for adjusting the standard deviation in a deterministic way such that the evolution strategy can converge to the global optimum.

$^{Schwefel(1981),Bäck(1996)}$. Parents are mutated using:

$$\sigma'(i) = \sigma(i) \times exp(\tau' N(0,1) + \tau N_i(0,1))$$
$$x'(i) = x(i) + N(0, \sigma'(i)),$$

where τ and τ' are proportionality constants that are defined in terms of n.

Also, modern evolution strategies allow the use of recombination (either *sexual*, when only 2 parents are involved, or *panmictic*, when more than 2 parents are involved in the generation of the offspring).

Some representative applications of evolution strategies are $^{Schwefel(1981),Bäck(1996),Fogel(1995)}$: nonlinear control, structural optimization, image processing and pattern recognition, biometrics, classification, network optimization, and airfoil design.

2.2. *Evolutionary programming*

Lawrence J. Fogel (see Fig. 3) introduced in the 1960s an approach called “evolutionary programming”, in which intelligence is seen as an adaptive behavior $^{Fogel(1966),Fogel(1999)}$. The original motivation of this paradigm was to solve prediction problems using finite state automata.

Evolutionary programming emphasizes the behavioral links between parents and offspring, instead of trying to emulate some specific genetic operators (as in the case of the genetic algorithm $^{Goldberg(1989),Mitchell(1996)}$).

Fig. 3. Lawrence J. Fogel. He developed an approach called "evolutionary programming", in which intelligence is seen as an adaptive behavior.

The basic algorithm of evolutionary programming is very similar to that of the evolution strategy. A population of individuals is mutated to generate a set of offspring. However, in this case, there are normally several types of mutation operators and no recombination (of any type), since evolution is modeled at the species level and different species do not interbreed. Another difference with respect to evolution strategies is that in this case, each parent produces exactly one offspring. Also, the decision of whether or not a parent will participate in the selection process is now determined in a probabilistic way, whereas in the evolution strategy this is a deterministic process. Finally, no encoding is used in this case (similarly to the evolution strategy) and emphasis is placed on the selection of the most appropriate representation of the decision variables.

In its original version, evolutionary programming didn't have a mechanism to self-adapt its parameters. However, in the early 1990s, Fogel and some of his co-workers realized of the importance of such mechanism and proposed the so-called "meta-evolutionary programming" $^{Fogeletal.(1991)}$ for continuous optimization. Over time, other self-adaptation mechanisms were proposed also for discrete optimization $^{Angelineetal.(1996)}$.

Interestingly, it was until the early 1990s that the evolution strategies community met the evolutionary programming community, despite the fact that both paradigms share very evident similarities $^{Bäck(1996),Fogel(1999)}$.

Some representative applications of evolutionary programming are $^{Fogel(1995),Fogel(1999)}$: forecasting, games, route planning, pattern recognition, and neural networks training.

2.3. *Genetic Algorithms*

Genetic algorithms (originally denominated "genetic reproductive plans") were introduced by John H. Holland (see Fig. 4) in the early 1960s [Holland(1962),Holland(1962a)]. The main motivation of this work was the solution of machine learning problems [Holland(1975)].

Fig. 4. John H. Holland. He introduced genetic algorithms.

Genetic algorithms emphasize the importance of sexual recombination (which is the main operator) over the mutation operator (which is used as a secondary operator). They also use probabilistic selection (like evolutionary programming and unlike evolution strategies). Before describing the way in which a genetic algorithm (GA) works, we will provide some of the basic terminology adopted by the researchers from this area [HeitkoetterandBeasley(1995)] in Tables 1 and 2.

The basic operation of a Genetic Algorithm is illustrated in the following segment of pseudo-code [BucklesandPetry(1992)]:

```
generate initial population, G(0);
evaluate G(0);
t:=0;
repeat
  t:=t+1;
  generate G(t) using G(t-1) (applying genetic operators);
  evaluate G(t);
until the stop condition is reached
```

First, an initial population is randomly generated. The individuals of

Table 1. Basic terminology of genetic algorithms.

Term	Definition
A **chromosome**	A data structure that holds a "string" of task parameters, or genes. This string may be stored, for example, as a binary bit-string (binary representation) or as an array of integers (floating point o real-coded representation) that represent a floating point number. This chromosome is analogous to the base-4 chromosomes present in our own DNA. Normally, in the GA community, the haploid model of a cell is assumed (one-chromosome individuals). However, diploids have also been used in the specialized literature $^{Goldberg(1989)}$.
A **gene**	A subsection of a chromosome that usually encodes the value of a single parameter (i.e., a decision variable).
An **allele**	The value of a gene. For example, for a binary representation each gene may have an allele of 0 or 1, and for a floating point representation, each gene may have an allele from 0 to 9.
A **schema**	A pattern of gene values in a chromosome, which may include "do not care" states (represented by a # symbol). Thus, in a binary chromosome, each schema can be specified by a string of the same length as the chromosome, with each character being one of { 0, 1, # }. A particular chromosome is said to "contain" a particular schema if it matches the schema (e.g. chromosome 01101 matches schema #1#0#).
The **fitness** of an individual	A value that reflects its performance (i.e., how well solves a certain task). A **fitness function** is a mapping of the chromosomes in a population to their corresponding fitness values. A **fitness landscape** is the hypersurface obtained by applying the fitness function to every point in the search space.
A **building block**	A small, tightly clustered group of genes which have co-evolved in such a way that their introduction into any chromosome will be likely to give increased fitness to that chromosome. The **building block hypothesis** $^{Goldberg(1989)}$ states that GAs generate their solutions by first finding as many building blocks as possible, and then combining them together to give the highest fitness.
Deception	A condition under which the combination of good building blocks leads to reduced fitness, rather than increased fitness. This condition was proposed by Goldberg as a reason for the failure of GAs on certain tasks $^{Goldberg(1989)}$.

this population will be a set of chromosomes or strings of characters (letters and/or numbers) that represent all the possible solutions to the problem.

One aspect that has great importance in the case of the genetic algorithm is the encoding of solutions. Traditionally, a binary encoding has been adopted, regardless of the type of decision variables of the problem to be solved $^{Goldberg(1989)}$. Holland provides some theoretical and biolog-

Table 2. Basic terminology of genetic algorithms (Continuation of Table 1).

Term	Definition
Elitism (or an elitist strategy)	A mechanism which ensures that the chromosomes of the highly fit member(s) of the population are passed on to the next generation without being altered by any genetic operator. The use of elitism guarantees that the maximum fitness of the population never decreases from one generation to the next, and it normally produces a faster convergence of the population. More important yet is the fact that it has been (mathematically) proven that elitism is necessary in order to be able to guarantee convergence of a simple genetic algorithm towards the global optimum $^{Rudolph(1994)}$.
Epistasis	The interaction between different genes in a chromosome. It is the extent to which the contribution to fitness of one gene depends on the values of other genes. Geneticists use this term to refer to a "masking" or "switching" effect among genes, and a gene is considered to be "epistatic" if its presence suppresses the effect of a gene at another locus (or position in the chromosome). This concept is closely related to deception, since a problem with high degree of epistasis is deceptive, because building blocks cannot be formed. On the other hand, problems with little or no epistasis are trivial to solve (hill climbing is sufficient).
Exploitation	The process of using information gathered from previously visited points in the search space to determine which places might be profitable to visit next. Hill climbing is an example of exploitation, because it investigates adjacent points in the search space, and moves in the direction giving the greatest increase in fitness. Exploitation techniques are good at finding local minima (or maxima). The GA uses crossover as an exploitation mechanism.
Exploration	The process of visiting entirely new regions of a search space, to see if anything promising may be found there. Unlike exploitation, exploration involves leaps into unknown regions. Random search is an example of exploration. Problems which have many local minima (or maxima) can sometimes only be solved using exploration techniques such as random search. The GA uses mutation as an exploration mechanism.
A **genotype**	A potential solution to a problem, and is basically the string of values chosen by the user, also called chromosome.
A **phenotype**	The meaning of a particular chromosome, defined externally by the user.
Genetic drift	The name given to the changes in gene/allele frequencies in a population over many generations, resulting from chance rather than from selection. It occurs most rapidly in small populations and can lead to some alleles to become extinct, thus reducing the genetic variability in the population.
A **niche**	A group of individuals which have similar fitness. Normally in multiobjective and multimodal optimization, a technique called **fitness sharing** is used to reduce the fitness of those individuals who are in the same niche, in order to prevent the population to converge to a single solution, so that stable sub-populations can be formed, each one corresponding to a different objective or peak (in a multimodal optimization problem) of the function $^{DebandGoldberg(1989)}$.

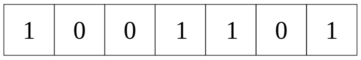


Fig. 5. Example of the binary encoding traditionally adopted with the genetic algorithm.

ical arguments for using a binary encoding $^{Holland(1975)}$. However, over the years, other types of encodings have been proposed, including the use of vectors of real numbers and permutations, which lend themselves as more "natural" encodings for certain types of optimization problems $_{Michalewicz(1996), Rothlauf(2002)}$.

Once an appropriate encoding has been chosen, we apply a *fitness function* to each one of these chromosomes in order to measure the quality of the solution encoded by the chromosome. Knowing each chromosome's fitness, a *selection* process takes place to choose the individuals (presumably, the fittest) that will be the parents of the following generation. The most commonly used selection schemes are described in Table 3 $^{GoldbergandDeb(1991)}$.

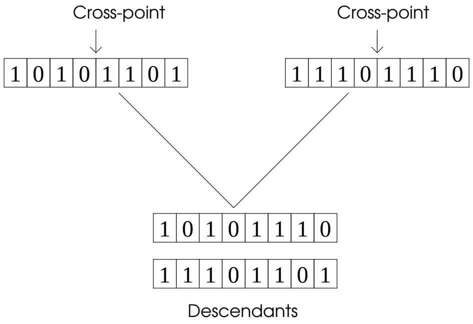


Fig. 6. Use of a single-point crossover between two chromosomes. Notice that each pair of chromosomes produces two descendants for the next generation. The cross-point may be located at the string boundaries, in which case the crossover has no effect and the parents remain intact for the next generation.

After being selected, *crossover* takes place. During this stage, the genetic material of a pair of individuals is exchanged in order to create the population of the next generation. There are three main ways of performing crossover:

Table 3. The basic selection schemes of genetic algorithms.

Scheme	Definition
Proportionate Selection	This term is used generically to describe several selection schemes that choose individuals for birth according to their objective function values f. In these schemes, the probability of selection p of an individual from the ith class in the tth generation is calculated as $p_{i,t} = \frac{f_i}{\sum_{j=1}^{k} m_{j,t} f_j}$ (1) where k classes exist and the total number of individuals sums to n. Several methods have been suggested for sampling this probability distribution, including Monte Carlo or *roulette wheel* selection [DeJong(1975)], *stochastic remainder* selection [Booker(1982),Brindle(1981)], and *stochastic universal* selection [Baker(1987),GrefenstetteandBaker(1989)].
Ranking Selection	In this scheme, proposed by Baker [Baker(1985)] the population is sorted from best to worst, and each individual is copied as many times as it can, according to a non-increasing assignment function, and then proportionate selection is performed according to that assignment.
Tournament Selection	The population is shuffled and then is divided into groups of k elements from which the best individual (i.e., the fittest) will be chosen. This process has to be repeated k times because on each iteration only m parents are selected, where $m = \frac{\textit{population size}}{k}$ For example, if we use binary tournament selection ($k = 2$), then we have to shuffle the population twice, since at each stage half of the parents required will be selected. The interesting property of this selection scheme is that we can guarantee multiple copies of the fittest individual among the parents of the next generation.
Steady State Selection	This is the technique used in Genitor [Whitley(1998)], which works individual by individual, choosing an offspring for birth according to linear ranking, and choosing the currently worst individual for replacement. In steady-state selection only a few individuals are replaced in each generation: usually a small number of the least fit individuals are replaced by offspring resulting from crossover and mutation of the fittest individuals. This selection scheme is normally used in evolving rule-based systems in which incremental learning (and remembering what has already been learned) is important and in which members of the population collectively (rather than individually) solve the problem at hand [Mitchell(1996)].

(1) *Single-point crossover*: A position of the chromosome is randomly selected as the crossover point as indicated in Fig. 6.

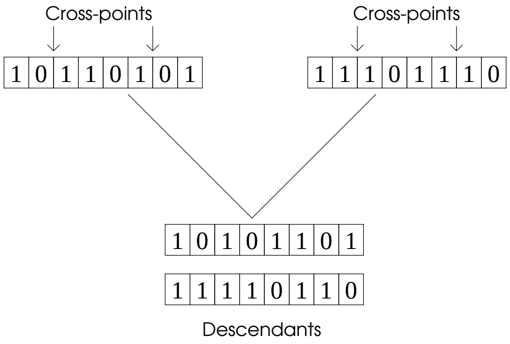


Fig. 7. Use of a two-point crossover between two chromosomes. In this case the genes at the extremes are kept, and those in the middle part are exchanged. If one of the two cross-points happens to be at the string boundaries, a single-point crossover will be performed, and if both are at the string boundaries, the parents remain intact for the next generation.

(2) *Two-point crossover*: Two positions of the chromosome are randomly selected as to exchange chromosomic material, as indicated in Fig. 7.

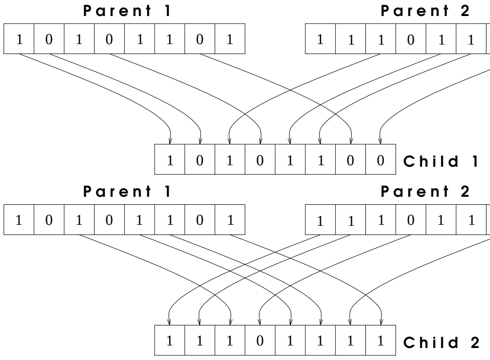


Fig. 8. Use of 0.5-uniform crossover (i.e., adopting a 50% probability of crossover) between two chromosomes. Notice how half of the genes of each parent go to each of the two children. First, the bits to be copied from each parent are selected randomly using the probability desired, and after the first child is generated, the same values are used to generate the second child, but inverting the source of procedence of the genes.

(3) *Uniform crossover*: This operator was proposed by Syswerda [Syswerda(1989)] and can be seen as a generalization of the two previ-

ous crossover techniques. In this case, for each bit in the first offspring it decides (with some probability p) which parent will contribute its value in that position. The second offspring would receive the bit from the other parent. See an example of 0.5-uniform crossover in Fig. 8. Although for some problems uniform crossover presents several advantages over other crossover techniques [Syswerda(1989)], in general, one-point crossover seems to be a bad choice, but there is no clear winner between two-point and uniform crossover [Mitchell(1996),Michalewicz(1996)].

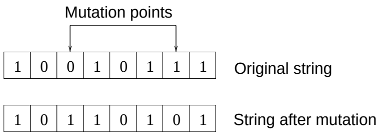


Fig. 9. An example of mutation using binary representation.

Mutation is another important genetic operator that randomly changes a gene of a chromosome. If we use a binary representation, a mutation changes a 0 to 1 and viceversa. An example of how mutation works is displayed in Fig. 9. This operator allows the introduction of new chromosomic material to the population and, from the theoretical perspective, it assures that—given any population—the entire search space is connected [BucklesandPetry(1992)].

If we knew in advance the final solution, it would be trivial to determine how to stop a genetic algorithm. However, as this is not normally the case, we have to use one of the two following criteria to stop the GA: either give a fixed number of generations in advance, or verify when the population has become homogeneous (i.e., all or most of the individuals have the same fitness).

Traditionally, genetic algorithms do not have a self-adaptation mechanism. Therefore, one of their main drawbacks is that their parameters tend to be fine-tuned in an empirical manner (i.e., by trial-and-error).

Some representative applications of genetic algorithms are the following [Goldberg(1989),Eiben(2003),Bäck(1996)]: data mining, optimization (structural, combinatorial, etc.), pattern recognition, and robot motion planning.

2.4. *Genetic programming*

One of the original goals of artificial intelligence (AI) was the automatic generation of computer programs that could produce a desired task given a certain input. During several years, such a goal seemed too ambitious since the size of the search space increases exponentially as we extend the domain of a certain program and, consequently, any technique will tend to produce programs that are either invalid or highly inefficient.

Fig. 10. John Koza. He suggested the use of a genetic algorithm with a tree-based encoding.

Some early evolutionary algorithms were attempted in automatic programming tasks, but they were unsuccessful and were severly criticized by some AI researchers $^{Fogel(1995)}$. Over the years, researchers realized that the key issue for using evolutionary algorithms in automatic programming tasks was the encoding adopted. In this regard, John Koza $^{Koza(1992)}$ (see Fig. 10) suggested the use of a genetic algorithm with a tree-based encoding. In order to simplify the implementation of such an approach, the original implementation of this sort of approach (which was called "genetic programming") was done under LISP, taking advantage of the fact that such programming language has a built-in parser.

The tree-encoding adopted by Koza obviously requires of different alphabets and specialized operators for evolving randomly generated programs until they become 100% valid. Note however, that the basic principles of this technique may be generalized to any other domain and, in fact, genetic programming has been used in a wide variety of applications $^{Koza(1992)}$.

The trees used in genetic programming consist of both functions and

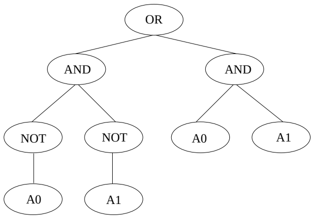


Fig. 11. An example of a chromosome used in genetic programming.

terminals. The functions normally adopted are the following [Koza(1992)]:

Arithmetic operations (e.g., +, -, ×, ÷)
Mathematical functions (e.g., sine, cosine, logarithms, etc.)
Boolean Operations (e.g., AND, OR, NOT)
Conditionals (IF-THEN-ELSE)
Loops (DO-UNTIL)
Recursive Functions
Any other domain-specific function

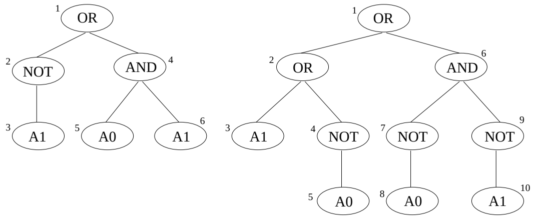


Fig. 12. The tree nodes are numbered before applying the crossover operator.

Terminals are typically variables or constants, and can be seen as functions that take no arguments. An example of a chromosome that uses the functions F={AND, OR, NOT} and the terminals T={A0, A1} is shown in Fig. 11.

Crossover can be applied by numbering the tree nodes corresponding to the two parents chosen (see Fig. 12) and (randomly) selecting a point in each of them such that the subtrees below such point are exchanged (see Fig. 13, where we assume that the crossover point for the first parent is 2 and for the second is 6). Typically, the sizes of the two parent trees will be different as in the example previously shown. It is also worth noticing that if the crossover point is the root of one of the parent trees, then the

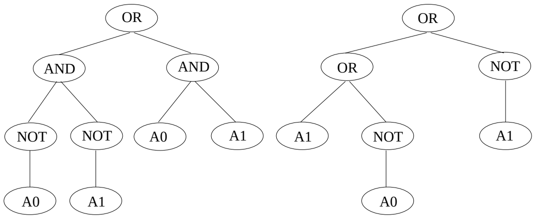
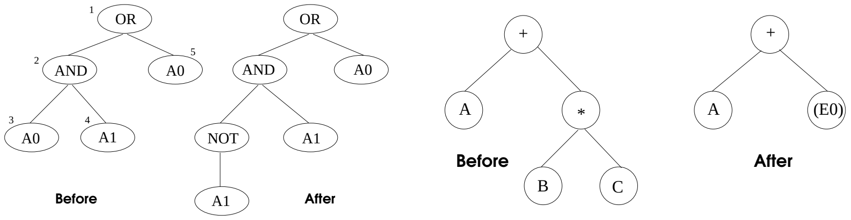


Fig. 13. The two offspring generated after applying the crossover operator.

whole chromosome will become a subtree of the other parent. This allows the incorporation of subroutines in a program. It is also possible that the roots of both parents are selected as crossover points. Should that be the case, the crossover operator will have no effect and the offspring will be identical to their parents.

Normally, genetic programming implementations impose a limit on the maximum depth that a tree can reach, as to avoid the generation (as a byproduct of crossover and mutation) of trees of very large size that could produce a memory overflow $^{Banzhafetal.(1998)}$.



(*a*) (*b*)

Fig. 14. An example of mutation in genetic programming (a) and of encapsulation in genetic programming (b).

Mutation in genetic programming takes place through a (random) selection of a certain node tree. The subtree below the chosen node is replaced by another tree which is randomly generated. Fig. 14a shows an example of the use of this operator (the mutation point in this example is node 3).

In genetic programming is also possible to protect or "encapsulate" a certain subtree which we know to contain a good building block, as to avoid that it is destroyed by the genetic operators. The selected subtree is replaced by a symbolic name that points to the real location of the subtree.

Such subtree is separately compiled and linked to the rest of the tree in an analogous way to the external classes of object oriented languages. Fig. 14b shows an example of encapsulation in which the right subtree is replaced by the name (**E0**).

Finally, genetic programming also provides mechanisms to destroy a certain percentage of the population such that we can renovate the chromosomic material after a certain number of generations. This mechanism, called **execution**, is very useful in highly complex domains in which our population may not contain a single feasible individual even after a considerably large number of generations.

3. A More General View of Evolutionary Algorithms

Despite the obvious differences and motivations of each of the aforementioned paradigms, the trend in the last few years has been to decrease the differences among the paradigms and refer (in generic terms) simply to evolutionary algorithms when talking about any of them.

In Table 4, the basic components to implement an EA in order to solve a problem [Michalewicz(1996),Eiben(2003)] are given. Also, it is shown that EAs differ from traditional search techniques in several ways [BucklesandPetry(1992)].

It is worth indicating that the ever-growing popularity of evolutionary algorithms in a variety of application domains tends to be related to their good reputation as "optimizers" (either for single-objective or for multi-objective problems [Osyczka(2002),CoelloCoelloetal.(2002)]). This is remarkable if we consider that some of them (namely, genetic algorithms) were not originally proposed for that type of application and that their use in optimization tasks has been questioned by some well-established researchers in the evolutionary computation community [DeJong(1993)]. Apparently, the reported success of evolutionary algorithms has resulted sufficiently convincing for practitioners and therefore their popularity [Bäcketal.(1997)].

4. Some Applications in Biometrics

Since their very inception, several researchers have attempted to use evolutionary algorithms for tasks related to pattern recognition, image processing, classification and machine learning [Cavicchio(1970),Cornett(1972),Trellue(1973),Fitzpatricketal.(1984),Stadnyk(1987),Wilson(1985),Englander(1985)].

However, most of this early work was mainly focused on the algorithmic development aspect rather than on an specific application. Thus, for some

Table 4. Basic components to implement an EA and particular features of an EA.

Requirements of an EA	Particular features of an EA
(1) A representation of the potential solutions to the problem. The selection of an appropriate encoding scheme tends to be crucial for the good performance of an EA [Rothlauf(2002)]. (2) A way to create an initial population of potential solutions (this is normally done randomly, but deterministic approaches can also be used). (3) An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness". The definition of a good fitness function is also vital for having a good performance. (4) A selection procedure that chooses the parents that will reproduce. (5) Evolutionary operators that alter the composition of children (normally, crossover and mutation). (6) Values for various parameters that the evolutionary algorithm uses (population size, probabilities of applying evolutionary operators, etc.).	• EAs do not require problem specific knowledge to carry out a search. However, if such knowledge is available, it can be easily incorporated as to make the search more efficient. • EAs use stochastic instead of deterministic operators and appear to be robust in noisy environments. • EAs are conceptually simple and easy to implement. • EAs have a wide applicability. • EAs are relatively simple to parallelize. • EAs operate on a population of potential solutions at a time. Thus, they are less susceptible to false attractors (i.e., local optima).

time, there was relatively little research on the development of evolutionary algorithms for specific biometric applications. This situation has changed in the last few years [Soldeketal.(1997)], although the use of other soft computing techniques such as neural networks is still more common than the use of evolutionary algorithms [HuangandYan(1997)].

Next, we will review a few case studies in this area which aim to provide a general picture of the type of research being conducted nowadays. A summary of these case studies is provided in Table 5.

4.1. *Fingerprint compression*

Grasemann and Miikulainen [GrasemannandMiikulainen(2005)] proposed a coevolutionary genetic algorithm based on a technique known as *Enforced Sub-Populations* [GomezandMiikulainen(1999)] and on a mathematical technique called *Lifting* to find wavelets that are specially adapted to a particular class

Table 5. Applications of evolutionary algorithms in biometrics.

Biometric	**Technique of EA**
lightgray.75 light	Physiological biometrics
Fingerprints	In [GrasemannandMiikulainen(2005)], a coevolutionary genetic algorithm proposed based on *Enforced Sub-Populations* technique [GomezandMiikulainen(1999)] and on a mathematical technique called *Lifting* to find wavelets that are specially adapted to a particular class of images. Other researchers have also worked on the use of evolutionary algorithms for solving other related problems such as optimizing the alignment of a pair of fingerprint images [Hanyetal.(2000)], estimation of the ridges in fingerprints [Ahmedetal.(1999)] and fingerprint classification [Qietal.(1998)].
Face	In [HoandHuang(2001)], a genetic algorithm-based approach have been proposed for facial modeling from an uncalibrated face image using a flexible generic parameterized facial model. Other researchers have also used evolutionary algorithms for related problems, such as human face detection [YokooandHagiwara(1996)], automated face recognition [TellerandVeloso(1995),Huang(1998)], and human posture estimation [Reindersetal.(1992)], among others.
Palmprints	In [Kharmaetal.(2003)], a cooperative coevolutionary genetic algorithm is used for hand-based feature selection. The outcome of the approach is a number of good (i.e., tight and well-separated) clusters which exist in the smallest possible feature space.
lightgray.75 light	Behavioral biometrics
Signature	In [?], genetic programming has been adopted for handwritten character recognition. Fitness function consists of the weighted sum of errors produced by each of the programs generated (zero-argument functions, which consists of the features to be extracted from the images that are presented as test cases). Each program also receives a penalty when there is a lack of diversity. Other researchers have also used evolutionary algorithms for related problems, such as unsupervised learning in handwritten character recognition tasks [Moritaetal.(2003)], design of a voting system to select from among multiple classifiers adopted for cursive handwritten text recognition [GünterandBunke(2004)], and interpretation of characters for ebooks [Leungetal.(2004)].
Keystroke dynamics	In [YuandCho(2004)], a combination of a support vector machine and a genetic algorithm is used for keystroke dynamics identity verification. The authors also propose the use of an ensemble model based on feature selection to alleviate the defficiencies of a small training data set.
Voice	In [Ganchevetal.(2004)] a locally recurrent probabilistic neural network was adopted for speaker verification. The approach is based on a three-step training procedure, from which the third step was done with differential evolution (an evolutionary algorithm that is highly competitive for numerical optimization tasks in which the decision variables are real numbers).

[1ex] height

of images. This approach was tested in the fingerprint domain against standard wavelets (including a winner of a competition held by the FBI in the early 1990s) to find the best wavelet fingerprint compression. The evolutionary algorithm outperformed the algorithm used by the FBI, which attracted a lot of attention from the media and from the evolutionary computation community, because this was a clear example of how an evolutionary design can outperform a human design.

Enforced sub-populations refers to an approach in which a number of populations of individual neurons (from a neural network) are evolved in parallel. In the evaluation phase, this approach repeatedly selects one neuron from each subpopulation to form candidate neural networks. The fitness of a particular neuron is the average fitness of all the neural networks in which it participated. This same idea was used in this work, but evolving different wavelets instead. These wavelets are constructed using the so-called *lifting step* $^{Sweldens(1996)}$, which is a finite filter that can generate new filter pairs from existing pairs.

To validate the approach, the authors used 80 available images. The size of each image was 300 by 300 pixels, at 500 dpi resolution. Each of them was used once as a test image and 79 times as part of the training set. In the experiments performed, the best wavelet found after each generation was used to compress the test image. It turns out that after only 50 generations, the evolutionary algorithm was able to produce results that outperformed the wavelets used by the FBI. Overall, the evolutionary algorithm introduced between a 15% and a 20% decrease in the space requirements for the same image quality. This is very significant if we consider that the FBI has over 50 million fingerprint cards on file and has recently started to store them electronically (the FBI is currently digitizing and compressing between 30,000 and 50,000 new cards every day).

4.2. *Facial modeling*

Ho and Huang $^{HoandHuang(2001)}$ proposed a genetic algorithm-based approach for facial modeling from an uncalibrated face image using a flexible generic parameterized facial model (FGPFM). The authors adopted the so-called "Intelligent Genetic Algorithm" IGA $^{Hoetal.(1999)}$ (which they had previously proposed) to tackle this problem. The approach of the authors consists in:

(*a*) Considering that the microstructural information can be expressed using the structural FGPFM with representative facial features that can be found in the image.

(*b*) The reconstruction procedure is considered as a block function of FGPFM where the input parameters are the 3D face-centered coordinates of a set of control points.

(*c*) Once these control points are given, the 3D facial model that we aim to find is determined based on the topological and geometrical descriptions of FGPFM.

So, the problem of reconstructing the 3D facial model is transformed into the problem of

> *How to acquire 3D control points that are sufficiently accurate as to provide the desired model.*

This gives raise to a large parameter optimization problem that is solved using the IGA. The fitness function adopted considers two criteria for evaluating the quality of the facial model:

(*a*) The projection of the facial model from some viewpoint must coincide with the features in the given face image, and

(*b*) The facial model must adhere to the generic knowledge of human faces accepted by the human perception.

An interesting aspect of this work is that the authors adopt a coarse-to-fine approach to solve the problem. This refers to a solution process that consists of several stages and that combines:

A *global* (coarse) search, and
A *local* (fine) search.

The approach was validated by performing several experiments in which synthetic face images and actual face images were analyzed. The results showed that the approach was robust and provided very good results with respect to other techniques based on genetic algorithms.

4.3. *Hand Image Classification*

Kharma et al. $^{Kharmaetal.(2003)}$ used a genetic algorithm with a cooperative co-evolutionary scheme for dynamic clustering and feature selection. Cooperative coevolution refers to the simultaneous evolution of two populations whose fitnesses are coupled (i.e., the fitness of one population depends on the fitness of the other one). The authors use this sort of scheme to integrate dynamic clusting with hand-based feature selection. For the feature extraction task, the authors considered two types of features: (1) geometric (finger width, finger height, finger circumference, finger angle, finger base length and plam aspect ratio), and (2) statistical (central moments, Fourier descriptors and Zernike moments). The authors optimized three quantities: (1) the set of features used for clustering, (2) the cluster centers, and (3) the total number of clusters. In the scheme adopted, a number of cluster centers are initially proposed. Then, a point (input pattern) is assigned to the cluster whose center is closest to the point, and the points are re-assigned to the new clusters based on their distance from the new cluster centers. This process is repeated until the locations of the cluster centers stabilize. Two populations are adopted: one of cluster centers and another one of dimension selections). During the process, the less discriminatory features are eliminated, which leaves a more efficient subset for further use. So, the outcome of the approach is a number of good (i.e., tight and well-separated) clusters which exist in the smallest possible feature space.

In order to validate this approach, the authors used 100 hand images and 84 normalized features. The results were found to be quite promising, since the average hand misplacement rate was 0.0580 with a standard deviation of 2.044, and half of the original 84 features adopted were eliminated during the evolutionary process. However, results were not compared to any other type of approach.

4.4. *Handwritten character recognition*

Teredesai et al. [?] adopted genetic programming in handwritten character recognition. Traditionally, active pattern recognition involves the use of an active heuristic function (e.g., a neural network $^{DudaandHart(1973)}$) which adaptively determines:

The *length* of the feature vector, and
The *features* themselves

used to classify an input pattern. The outputs of this stage are:

The *confidence* values and
The *separability* values.

Then, in a postprocessing stage, decision making takes place, based on the output from the classification stage. This is followed by an iterative search process in which sub images of finer resolution are fed to the classification process until reaching a level of confidence that is satisfactory (or until we run out of time). Contrasting this traditional process, Teredesai et al. [?] proposed to eliminate the iterative search process and focused instead on searching for the areas in the image that have maximum separability information. They proposed two possible ways of achieving this goal:

(*a*) Use only those features from the feature set that provide maximum separability, and
(*b*) Generate a new set of features that represents the image data in a manner that provides a better classification accuracy.

These two tasks are tightly coupled and the authors propose to use genetic programming to decouple them. The most interesting aspect of the proposal is the fitness function adopted. Such fitness function consists of the weighted sum of errors produced by each of the programs generated. The "programs" are really zero-argument functions, which consist of the features to be extracted from the images that are presented as test cases. Each program also receives a penalty when there is a lack of diversity (i.e., poor accuracy of the classifiers is penalized). So, the authors were evolving features that provided maximum separability, but at the same time, were trying to achieve (through the penalty function adopted) a better classification accuracy.

The proposed approach was validated using several datasets which included the NIST handwritten digit sets, and the CEDAR-Digit data set (which has more noisy data). The NIST data set consists of 159,228 images in the training set, and 53,300 images in the test set. The CEDAR-digit data set consists of 7,245 images in the training set and 2,711 images in the test set. The CEDAR-digit data set is more challenging, because it contains more noisy data with images that were incorrectly recognized or rejected even by the current recognizers based on the K-nearest neighbor rule and neural networks. The proposed approach was compared with respect to a traditional passive classifier. The results indicated that the GP-based approach was able to outperform the traditional passive classifier, achieving an accuracy of 97.1% using a very low number of features.

4.5. *Keystroke Dynamics Identity Verification*

Yu and Cho $^{YuandCho(2004)}$ used a combination of a support vector machine (SVM) and a genetic algorithm (GA) for keystroke dynamics identity verification. It has been known at least since the 1980s that a user's keystroke pattern is highly repeatable and distinct from that of other users

$^{Gainesetal.(1980)}$. Thus, it is possible to collect a user's timing vectors and use them to build a model that discriminates the owner from possible imposters.

The authors adopted a SVM for novelty detection and a GA-wrapper feature selection approach to automatically select a relevant subset of features, while ignoring the rest. The authors argue that the use of a SVM only requires about 1/1000 of the training time that a neural network would require. The authors also propose the use of an ensemble model based on feature selection to alleviate the defficiencies of a small training data set. A wrapper approach tries many different sets of features by building a model using a learning algorithm, and then chooses the one with the best performance measure. Feature subset selection is basically an optimization problem, and the GA is used for that sake. On the other hand, the SVM is adopted as the base learner for the wrapper approach. The fitness function of the GA combines three different criteria: (1) the accuracy of the novelty detector, (2) the learning time and (3) the dimension reduction ratio. Thus, the GA deals with a population of SVMs which employ different feature subsets. At early stages of the evolutionary process, the candidate solutions or SVMs usually show a high level of diversity, but a low accuracy. At later stages of the search, the behavior is the opposite: there is less diversity and a higher accuracy. In order to deal with the problem of having insufficient data (something common within a keystroke dynamics identity verification system), the authors made a trade-off between diversity and accuracy by selecting diverse candidates from the population immediately before it converges. Such candidates are used to create an ensemble.

The proposed approach was validated using 21 different password-typing patterns and results were compared with respect to the use of neural networks, two other SVM-based techniques and a random feature ensemble approach. Although the proposed approach presented a similar performance as the others, its time requirements were considerably lower.

4.6. *Voice Recognition*

Ganchev et al. $^{Ganchevetal.(2004)}$ used a locally recurrent probabilistic neural network for the process of speaker verification. The use of a locally recurrent probabilistic scheme aims to exploit the inter-frame correlation among the feature vectors extracted from successive speech frames. The approach is based on a three-step training procedure. The first training step creates the actual topology of the network. The second training step involves the computation of a smoothing parameter for each class. In the third step, the weights of the locally recurrent layer are computed. For this sake, a type of evolutionary algorithm called "differential evolution" $^{Price(1999)}$ is adopted (this is done by minimizing an error function).

The proposed approach was validated using a text-independent speaker verification system previously developed by the same authors, which uses probabilistic neural networks $^{Ganchevetal.(2002)}$. Thus, the aim was to assess the improvements introduced by the new approach. The authors used data from POLYCOST (a telephone-speech database for speaker recognition) $^{Hennebertetal.(2000)}$ for their study. The results indicated that the proposed approach presented a relative reduction of the error rate of more than 28% with respect to the original scheme.

5. Some possible research directions

Despite the evident popularity of approaches such as statistical methods and other soft computing techniques (e.g., neural networks) in biometrics $^{Kungetal.(2004)}$, the use of evolutionary algorithms has attracted a growing interest in the last few years. However, there are still many possible research directions that may be worth exploring in this area. A few of them are briefly discussed in Tables 6 and 7.

6. Conclusions

This chapter has provided a short (and very general) introduction to evolutionary algorithms, including the historical roots of its main paradigms, their original motivation and some of their applications. Then, we analyzed a few case studies on the use of evolutionary algorithms to solve problems in biometrics.

In the final part of the chapter, we provided some of the possible research paths that we consider that could be pursued within the next few years. A wide variety of tasks involved in biometrics tasks can be solved (and have been solved in the past) using evolutionary algorithms (e.g., classification, feature extraction and pattern recognition), which opens a lot of research paths that have been only scarcely explored. Thus, we believe that the use of evolutionary algorithms (perhaps in combination with other soft computing techniques) in biometrics will significantly increase within the next few years.

Table 6. Possible research directions in the use of evolutionary algorithms in biometrics

Direction	Strategy
Multiobjective Optimization	Multiobjective optimization refers to the simultaneous optimization of two or more objective functions which are commonly in conflict with each other [CoelloCoelloetal.(2002)]. Due to the multiobjective nature of many real-world problems, it is very likely that the use of multiobjective optimization techniques becomes more popular in biometrics [Moritaetal.(2003)]. Among the possible techniques to solve multiobjective optimization problems, evolutionary algorithms present several advantages over traditional mathematical programming techniques [Miettinen(1998),Ehrgott(2005)]. For example, evolutionary algorithms have a population-based nature which allows them to generate several elements of the Pareto optimal set in one run. Additionally, evolutionary algorithms do not require an initial search point (as is the case of most mathematical programming techniques) and are less sensitive to the shape and continuity of the Pareto front [CoelloCoelloetal.(2002),Deb(2001)].
Use of Genetic Programming	As we saw before, the use of tree-encodings in a genetic algorithm (the so-called "genetic programming") is a powerful aid for automated programming. Such type of encoding is also very useful for classification and data mining tasks [Freitas(1997),Teredasaietal.(2001)]. Consequently, the use of genetic programming in biometrics, although not widely spread yet, is expected to considerably grow in the next few years. It is worth indicating, however, that the use of a more complex encoding, while allowing tackling more complex problems, also involves a higher computational cost (this applies to multiobjective optimization as well). Thus, the use of parallel computing seems an obvious choice in these cases [Talay(2005)].

Acknowledgments

The author acknowledges support from CONACyT through project No. 45683-Y.

References

Ahmed et al. (1999). Ahmed, S. Abutaleb, and Kamel, M. (1999). A genetic algorithm for the estimation of ridges in fingerprints, *IEEE Trans. Image Processing*, **8**, 8, pp. 1134–1139.

Hany et al. (2000). Hany, T. Ammar amd Yongyi Tao (2000). Fingerprint registration using genetic algorithms, *Proc. 3rd IEEE Symp. on Application-Specific Systems and Software Engineering Technology (ASSET'00)*, pp. 148–154, Richardson, Texas.

Angeline et al. (1996). Angeline, P. J., Fogel, D. B. and Fogel, L. J. (1996). A comparison of self-adaptation methods for finite state machines in a dynamic

Table 7. Possible research directions in .....(Continuation of Table 6).

Direction	Strategy
Use of Alternative Metaheuristics	In recent years, other biologically-inspired metaheuristics have become increasingly popular in a wide variety of applications [Corneetal.(1999)]. It is expected that several of these approaches are eventually adopted in biometric applications. Representative examples of these new metaheuristics are the following: (a) **Particle Swarm Optimization**: Proposed by Kennedy and Eberhart [KennedyandEberhart(1995),Kennedy(2001)], this metaheuristic simulates the movements of a group (or population) of birds which aim to find food. The approach can be seen as a distributed behavioral algorithm that performs (in its more general version) multidimensional search. In the simulation, the behavior of each individual is affected by either the best local (i.e., within a certain neighborhood) or the best global individual. The approach uses then the concept of population and a measure of performance similar to the fitness value adopted with evolutionary algorithms. The approach introduces the use of flying potential solutions through hyperspace (used to accelerate convergence) and allows individuals to benefit from their past experiences. This technique has been successfully used for both continuous nonlinear and discrete binary optimization [EberhartandShi(1998),KennedyandEberhart(1997),Kennedy(2001)] (b) **Artificial Immune Systems**: Computationally speaking, our immune system can be seen as a highly parallel intelligent system that is able to learn and retrieve previous knowledge (i.e., it has "memory") to solve recognition and classification tasks. Due to these interesting features, several researchers have developed computational models of the immune system and have used it for a variety of tasks, including classification and pattern recognition [Dasgupta(1999),NunesdeCastroandTimmis(2002),NunesdeCastroandVonZuben(2002)]. (c) **The Ant System**: This is a metaheuristic inspired by colonies of real ants, which deposit a chemical substance on the ground called *pheromone* [DorigoandDiCaro(1999),Colornietal.(1992),Dorigoetal.(1996),DorigoandStützle(2004)]. This substance influences the behavior of the ants: they tend to take those paths where there is a larger amount of pheromone. Pheromone trails can thus be seen as an indirect communication mechanism among ants. From a computer science perspective, the ant system is a multi-agent system where low level interactions between single agents (i.e., artificial ants) result in a complex behavior of the entire ant colony. The ant system was originally proposed for the traveling salesman problem (TSP), and most of the current applications of the algorithm require the problem to be reformulated as one in which the goal is to find the optimal path of a graph. A way to measure the distances between nodes is also required in order to apply the algorithm [Dorigo(1991)]. Despite this limitation, this approach has been found to be very successful in a variety of combinatorial optimization problems [DorigoandDiCaro(1999),Bonabeuetal.(1999),DorigoandStützle(2004)].

environment. In: L.J. Fogel, P.J. Angeline, and T. Bäck, Eds., *Evolutionary Programming V*, pp. 441–449, Cambridge, Massachusetts, MIT Press.

Bäck (1996). Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York.

Bäck et al. (1997). Bäck, T., Fogel, D.B., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation.* Institute of Physics Publishing and Oxford University Press, New York.

Baker (1985). Baker, J.E. (1985). Adaptive Selection Methods for Genetic Algorithms. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pp. 101–111. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Baker (1987). Baker, J.E. (1987). Reducing Bias and Inefficiency in the Selection Algorithm. In J.J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–22. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Banzhaf et al. (1998). Banzhaf, W., Nordin, P., Keller, R.E. and Fancone, F.D. (1998) *Genetic Programming. An Introduction.* Morgan Kaufmann Publishers, San Francisco, California.

Bonabeu et al. (1999). Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence. From Natural to Artificial Systems.* Oxford University Press, New York.

Booker (1982). Booker, L.B. (1982). *Intelligent Behavior as an Adaptation to the Task Environment.* PhD thesis, Logic of Computers Group, University of Michigan, Ann Arbor, Michigan.

Brindle (1981). Brindle, A. (1981). *Genetic Algorithms for Function Optimization.* PhD thesis, Department of Computer Science, University of Alberta, Edmonton, Alberta.

Buckles and Petry (1992). Buckles, B.P. and Petry, F.E. editors (1992). *Genetic Algorithms.* Technology Series. IEEE Computer Society Press.

Cannon (1932). Cannon, W.D. *The Wisdom of the Body.* Norton and Company, New York.

Cavicchio (1970). Cavicchio, D.J. (1970). *Adaptive Search Using Simulated Evolution.* PhD thesis, University of Michigan, Ann Arbor, Michigan.

Coello Coello et al. (2002). Coello Coello, C.A., Van Veldhuizen, D.A., and Lamont, G.B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, New York. ISBN 0-3064-6762-3.

Colorni et al. (1992). Colorni, A., Dorigo, M., and Maniezzo, V. (1992). Distributed Optimization by Ant Colonies. In F.J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, pp. 134–142. MIT Press, Cambridge, MA, USA.

Corne et al. (1999). Corne, D., Dorigo, M. and Glover, F. editors. (1999). *New Ideas in Optimization.* McGraw-Hill, London.

Cornett (1972). Cornett, F.N. (1972). An Application of Evolutionary Programming to Pattern Recognition. Master's thesis, New Mexico State University, Las Cruces, New Mexico, USA.

Darwin (1882). Darwin, C.R. (1882). *The Variation of Animals and Plants under*

Domestication. Murray, London, second edition.

Dasgupta (1999). Dasgupta, D., editor. (1999). *Artificial Immune Systems and Their Applications.* Springer-Verlag, Berlin.

Dawkins (1990). Dawkins, R. (1990). *The Blind Watchmaker.* Gardners Books. ISBN 0-140-14481-1.

Deb (2001). Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, Chichester, UK. ISBN 0-471-87339-X.

Deb and Goldberg (1989). Deb, K. and Goldberg, D.E. (1989). An Investigation of Niche and Species Formation in Genetic Function Optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 42–50, Morgan Kaufmann Publishers, San Mateo, California, USA.

Dorigo (1991). Dorigo, M., Maniezzo, V. and Colorni, A. (1991). Positive Feedback as a Search Strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo et al. (1996). Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, **26**(1):29–41.

Dorigo and Di Caro (1999). Dorigo, M. and Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pp. 11–32, McGraw-Hill, London, UK.

Dorigo and Stützle (2004). Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization.* The MIT Press, Cambridge, Massachusetts, USA. ISBN 0-262-04219-3.

Duda and Hart (1973). Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* Wiley International.

Eberhart and Shi (1998). Eberhart, R.C. and Shi, Y. (1998). Comparison between Genetic Algorithms and Particle Swarm Optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A.E. Eibe, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 611–619. Springer-Verlag.

Ehrgott (2005). Ehrgott, M. *Multicriteria Optimization.* Springer, Berlin, second edition. ISBN 3-540-21398-8.

Eiben (2003). Eiben, A.E. and Smith, J.E. (2003). *Introduction to Evolutionary Computing.* Springer, Berlin, Germany. ISBN 3-540-40184-9.

Englander (1985). Englander, A.C. (1985). Machine Learning of Visual Recognition Using Genetic Algorithms. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp. 197–201, Lawrence Erlbraum Associates, Hillsdale, New Jersey, USA.

Fitzpatrick et al. (1984). Fitzpatrick, J.M., Grefenstette, J.J. and Van Gutch, D. (1984). Image registration by genetic search. In *Proceedings of the IEEE Southeast Conference*, pp. 460–464.

Fogel (1995). Fogel, D.B. (1995). *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence.* The Institute of Electrical and Electronic Engineers, New York, USA.

Fogel (1998). Fogel, D.B., editor, (1998). *Evolutionary Computation. The Fos-*

sil Record. Selected Readings on the History of Evolutionary Algorithms. The Institute of Electrical and Electronic Engineers, New York, USA.

Fogel et al. (1991). Fogel, D.B., Fogel, L.J. and Atmar, J.W. (1991). Meta-evolutionary programming. In R.R. Chen, editor, *Proceedings of the 25th Asimolar Conference on Signals, Systems, and Computers*, pp. 540–545, IEEE Computer Society, Los Alamitos, California, USA.

Fogel (1966). Fogel, L.J. (1966). *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, USA.

Fogel (1999). Fogel, L.J. (1999). *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming.* John Wiley & Sons, Inc., New York, USA.

Freitas (1997). Freitas, A.A. (1997). A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction. In J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Proceedings of the Second Annual Conference on Genetic Programming*, pp. 96–101, Morgan Kaufmann Publishers, San Francisco, California, USA.

French (1994). French, M. (1994). *Invention and Evolution. Design in Nature and Engineering.* Cambridge University Press, UK, second edition. ISBN 0-521-46503-6.

Ganchev et al. (2002). Ganchev, T., Fakotakis, N. and Kokkinakis, G. (2002). Text-Independent Speaker Verification Based on Probabilistic Neural Networks. In *Proceedings of the Acoustics 2002*, pp. 159–166, Patras, Greece.

Ganchev et al. (2004). Ganchev, T., Tasoulis, D.K., Vrahatis, M.N. and Fakotakis, N. (2004). Locally Recurrent Probabilistic Neural Networks with Application to Speaker Verification. *GESTS International Transaction on Speech Science and Engineering*, **1**(2), pp. 1–13.

Gaines et al. (1980). Gaines, R., Lisowsky, W., Press, S., and Shapiro, N. (1980). Authentication by Keystroke Timing: Some Preliminary Results. Technical Report No. R-2526-NSF, Rand Corporation, Santa Monica, California, USA.

Goldberg (1989). Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co., Reading, Massachusetts, USA.

Goldberg and Deb (1991). Goldberg, D.E. and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pp. 69–93. Morgan Kaufmann, San Mateo, California, USA.

Gomez and Miikulainen (1999). Gomez, F. and Miikkulainen, R. (1999). Solving non-markovian control tasks with neuroevolution. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1356–1361, AAAI, San Francisco, California, USA.

Grasemann and Miikulainen (2005). Grasemann, U. and Miikkulainen, R. (2005). Effective image compression using evolved wavelets. In H.-G. Beyer et al., editor, *2005 Genetic and Evolutionary Computation Conference (GECCO'2005)*, vol. 2, pp. 1961–1968, ACM Press, New York, USA.

Grefenstette and Baker (1989). Grefenstette, J.J. and Baker, J.E. (1989) How Genetic Algorithms work: A critical look at implicit parallelism. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 20–27, Morgan Kaufmann Publishers, San Mateo, California, USA.

Günter and Bunke (2004). Günter, S. and Bunke, H. (2004). Optimization of Weights in a Multiple Classifier Handwritten Word Recognition System Using a Genetic Algorithm. *Electronic Letters on Computer Vision and Image Analysis*, **3**(1):25–41.

Heitkoetter and Beasley (1995). Heitkoetter, J. and Beasley, D. (1995). The hitch-hiker's guide to evolutionary computation (faq in comp.ai.genetic). USENET. (Version 3.3).

Hennebert et al. (2000). Hennebert, J., Melin, H., Petrovska, D., and Genoud, D. (2000). POLYCOST: A Telephone-Speech Database for Speaker Recognition. *Speech Communication*, **31**, pp. 265–270.

Ho and Huang (2001). Ho, S.-Y. and Huang, H.-L. (2001). Facial modeling from an uncalibrated face image using a coarse-to-fine genetic algorithm. *Pattern Recognition*, **34**:1015–1031.

Ho et al. (1999). Ho, S.-Y., Shu, L.-S., and Chen, H.-M. (1999). Intelligent Genetic Algorithm with a New Intelligent Crossover Using Orthogonal Arrays. In *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Vol. 1, pp. 289–296, Morgan Kaufmann Publishers, San Francisco, California, USA.

Holland (1962). Holland, J.H. (1962). Concerning efficient adaptive systems. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, editors, *Self-Organizing Systems—1962*, pp. 215–230. Spartan Books, Washington, D.C., USA.

Holland (1962a). Holland, J.H. (1962a). Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, **9**:297–314.

Holland (1975). Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, Michigan, USA.

Huang and Yan (1997). Huang, K. and Yan, H. (1997). Off-Line Signature Verification by a Neural Network Classifier. In *Proceedings of the 17th Australian Conference on Neural Networks*, pp. 190–194, Australian National University, Canberra, Australia.

Huang (1998). Huang, R.J. (1998). *Detection Strategies for Face Recognition Using Learning and Evolution.* PhD thesis, Department of Computer Science, George Mason University, Fairfax, Virginia, USA.

De Jong (1975). De Jong, K.A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, USA.

De Jong (1993). De Jong, K.A. (1993). Genetic Algorithms are NOT Function Optimizers. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pp. 5–17. Morgan Kaufmann Publishers, San Mateo, California, USA.

Kennedy and Eberhart (1995). Kennedy, J. and Eberhart, R.C. (1995). Particle Swarm Optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, IEEE Service Center, Piscataway, New Jersey, USA.

Kennedy and Eberhart (1997). Kennedy, J. and Eberhart, R.C. (1997). A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the 1997 IEEE Conference on Systems, Man, and Cybernetics*, pp. 4104–4109, IEEE Service Center, Piscataway, New Jersey, USA.

Kennedy (2001). Kennedy, J. and Eberhart, R.C. (2001). *Swarm Intelligence.* Morgan Kaufmann Publishers, San Francisco, California, USA.

Kharma et al. (2003). Kharma, N., Suen, C.Y. and Guo, P.F. (2003). Palm-Prints: A Novel Co-evolutionary Algorithm for Clustering Finger Images. In E. Cantú-Paz (editor) *2003 Genetic and Evolutionary Computation Conference (GECCO 2003)*, pp. 322–331, Springer-Verlag, Lecture Notes in Computer Science Vol. 2723, Chicago, Illinois, USA.

Koza (1992). Koza, J.R. (1992). *Genetic Programming. On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, USA.

Kung et al. (2004). Kung, S.Y., Mak, W.M. and Lin, S.H. (2004). *Biometric Authentication: A Machine Learning Approach.* Prentice Hall, USA. ISBN 0-1314-782-49.

Leung et al. (2004). Leung, K.F., Leung, F.H.F., Lam, H.K. and Ling, S.H. (2004). On interpretation of graffiti digits and characters for ebooks: neural-fuzzy network and genetic algorithm approach. *IEEE Transactions on Industrial Electronics*, **51**(2):464–471.

Michalewicz (1996). Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, New York, USA, third edition.

Miettinen (1998). Miettinen, K.M. (1998). *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, Boston, Massachusetts, USA.

Mitchell (1996). Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, Massachusetts, USA.

Morita et al. (2003). Morita, M.E., Sabourin, R., Bortolozzi, F., and Suen, C.Y. (2003). Unsupervised feature selection using multi-objective genetic algorithm for handwritten word recognition. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'2003)*, pp. 666–670, Edinburgh, Scotland.

Nunes de Castro and Timmis (2002). Nunes de Castro, L. and Timmis, J. (2002). *Artificial Immnue System: A New Computational Intelligence Approach.* Springer Verlang, Great Britain. ISBN 1-8523-594-7.

Nunes de Castro and Von Zuben (2002). Nunes de Castro, L. and Von Zuben, F.J. (2002). Learning and Optimization Using the Clonal Selection Principle. *IEEE Transactions on Evolutionary Computation*, **6**(3):239–251.

Osyczka (2002). Osyczka, A. (2002). *Evolutionary Algorithms for Single and Multicriteria Design Optimization.* Physica Verlag, Germany. ISBN 3-7908-1418-0.

Price (1999). Price, K.V. An Introduction to Differential Evolution. In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*, pp. 79–108, McGraw-Hill, London, UK.

Qi et al. (1998). Qi, Y., Tian, J. and Dai, R.-W. (1998). Fingerprint classification system with feedback mechanism based on genetic algorithm. In *Proceedings of the Fourteenth International Conference on Pattern Recognition*, Vol. 1, pp. 163–165, IEEE.

Rao (1996). Rao, S.S. (1996). *Engineering Optimization. Theory and Practice.*

John Wiley & Sons, Inc., third edition.

Rechenberg (1973). Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann–Holzboog, Stuttgart, Germany.

Reinders et al. (1992). Reinders, M.J.T., Sankur, B., and van der Lubbe, J.C.A. (1992). Tranformation of a general 3d facial model to actual scene face. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition*, Vol. III, Conference C: Image, Speech and Signal Analysis, pp. 75–78.

Rothlauf (2002). Rothlauf, F. (2002). *Representations for Genetic and Evolutionary Algorithms.* Physica-Verlag, New York, USA.

Rudolph (1994). Rudolph, G. (1994). Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks*, **5**:96–101.

Schwefel (1981). Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.* Wiley, Chichester, UK.

Soldek et al. (1997). Soldek, J., Shmerko, V., Phillips, P., Kukharev, G., Rogers, W., and Yanushkevich, S. (1997). Image Analysis and Pattern Recognition in Biometric Technologies. In *Proceedings of the International Conference on the Biometrics: Fraud Prevention, Enhanced Service*, pp. 270–286, Las Vegas, Nevada, USA.

Stadnyk (1987). Stadnyk, I. (1987). Schema Recombination in a Pattern Recognition Problem. In J.J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 27–35, Lawrence Erlbraum Associates, Hillsdale, New Jersey, USA.

Sweldens (1996). Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets. *Journal of Applied and Computational Harmonic Analysis*, **3**(2):186–120.

Syswerda (1989). Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 2–9, Morgan Kaufmann Publishers, San Mateo, California, USA.

Talay (2005). Talay, A.C. (2005). An Approach for Eye Detection Using Parallel Genetic Algorithm. In V.S. Sunderam, G.D. van Albada, P.M.A. Sloot, and J.J. Dongarra, editors, (2005). In *Computational Science ICCS 2005: 5th International Conference, Proceedings, Part III*, pp. 1004, Springer. Lecture Notes in Computer Science Vol. 3516, Atlanta, Georgia, USA.

Teller and Veloso (1995). Teller, A. and Veloso, M. (1995). Algorithm Evolution for Face Recognition: What Makes a Picture Difficult. In *International Conference on Evolutionary Computation (ICEC'95)*, pp. 608–613, IEEE, Perth, Australia.

Teredasai et al. (2001). Teredasai, A., Park, J., and Govindaraju, V. (2001). Active Handwritten Character Recognition Using Genetic Programming. In J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A.G.B. Tettamanzi, and W.B. Langdon, editors, (2001), *Genetic Programming. 4th European Conference, EuroGP 2001*, pp. 371–379, Springer, Lake Como, Italy.

Trellue (1973). Trellue, R.E. (1973). The Recognition of Handprinted Characters Through Evolutionary Programming. Master's thesis, New Mexico State University, Las Cruces, New Mexico, USA.

Whitley (1998). Whitley, D. (1989). The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116–121. Morgan Kaufmann Publishers, San Mateo, California, USA.

Wilson (1985). Wilson, S.W. (1985). Adaptive "Cortical" Pattern Recognition. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 188–196, Lawrence Erlbraum Associates, Hillsdale, New Jersey, USA.

Yokoo and Hagiwara (1996). Yokoo, Y. and Hagiwara, M. (1996). Human faces detection method using genetic algorithm. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 113–118, IEEE.

Yu and Cho (2004). Yu, E. and Cho, S. (2004). Keystroke dynamics identity verification-its problems and practical solutions. *Computers & Security*, **23**(5), pp. 428–440.

Zhang (2000). Zhang, D.D. (2000). *Automated Biometrics: Technologies and Systems.* Kluwer Academic Publishers, The Netherlands.