# Evolutionary computation in medicine: an overview

## Carlos Andrés Peña-Reyes *, Moshe Sipper

*Logic Systems Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, CH-1015 Lausanne, Switzerland*

**Abstract**

The term evolutionary computation encompasses a host of methodologies inspired by natural evolution that are used to solve hard problems. This paper provides an overview of evolutionary computation as applied to problems in the medical domains. We begin by outlining the basic workings of six types of evolutionary algorithms: genetic algorithms, genetic programming, evolution strategies, evolutionary programming, classifier systems, and hybrid systems. We then describe how evolutionary algorithms are applied to solve medical problems, including diagnosis, prognosis, imaging, signal processing, planning, and scheduling. Finally, we provide an extensive bibliography, classified both according to the medical task addressed and according to the evolutionary technique used. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Evolutionary computation; Genetic algorithms; Medical applications

*Evolution is the natural way to program.*

Thomas Ray

* Corresponding author. Tel.: + 41-21-693-2658; fax: + 41-21-693-3705.
*E-mail addresses:* carlos.pena@epfl.ch (C.A. Peña-Reyes), moshe.sipper@epfl.ch (M. Sipper)

## 1. Introduction

The Merriam–Webster OnLine Dictionary (www.m-w.com) defines medicine as "the science and art dealing with the maintenance of health and the prevention, alleviation, or cure of disease". In aiming to fulfill their defined mission medical professionals are confronted daily with problems from diverse walks of (medical) life, which all exhibit an underlying commonality: searching for a good solution among a (usually huge) space of possible solutions. Whether trying to pry out signs of malignant cancer in cell biopsies or looking for irregularities in EEG signals, the basic problem is that of sifting through a welter of candidate solutions to find as good a solution as possible.

As in any other area of modern life, computers are omnipresent in medicine, from the hospital accounting computer to the high-end MRI scanner. In particular computers are used as tools to abet medical professionals in resolving search problems. Numerous techniques have been applied over the past few decades to solve medical problems: expert systems, artificial neural networks, linear programming, and database systems are but a sampling of the approaches used. One of the relative newcomers in medicine is the approach known as evolutionary computation, which is the object of focus in this paper.

The idea of applying the biological principle of natural evolution to artificial systems, introduced more than four decades ago, has seen impressive growth in the past few years. Known as evolutionary algorithms or evolutionary computation, these techniques are common nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, circuit design, machine learning, economics, ecology, and population genetics, to mention but a few. In particular, evolutionary algorithms have been applied to problems in medicine.

This paper provides an overview of evolutionary computation in medicine. In the next section we delineate the basic workings of six types of evolutionary algorithms: genetic algorithms, genetic programming, evolution strategies, evolutionary programming, classifier systems, and hybrid systems. Section 2 thus provides a brief summary of how evolutionary computation works for readers unfamiliar with the approach. Section 3 then describes how evolutionary algorithms are applied to solve medical problems, including diagnosis, prognosis, imaging, signal processing, planning, and scheduling. Finally, Section 4 provides an extensive bibliography, classified both according to the medical task addressed and according to the evolutionary technique used.

## 2. Evolutionary computation

The domain of evolutionary computation involves the study of the foundations and the applications of computational techniques based on the principles of natural evolution. Evolution in nature is responsible for the 'design' of all living beings on earth, and for the strategies they use to interact with each other. Evolutionary

algorithms employ this powerful design philosophy to find solutions to hard problems.

Generally speaking, evolutionary techniques can be viewed either as search methods, or as optimization techniques. As written by Michalewicz [66]: ''Any abstract task to be accomplished can be thought of as solving a problem, which, in turn, can be perceived as a search through a space of potential solutions. Since usually we are after 'the best' solution, we can view this task as an optimization process''.

The first works on the use of evolution-inspired approaches to problem solving date back to the late 1950s [13,14,28,33,34]. Independent and almost simultaneous research conducted by Rechenberg and Schwefel on evolution strategies [81,82,86,87], by Holland on genetic algorithms [44,46], and by Fogel on evolution- ary programming [31,32] triggered the study and the application of evolutionary techniques.

Three basic mechanisms drive natural evolution: *reproduction*, *mutation*, and *selection*. These mechanisms act ultimately on the *chromosomes* containing the genetic information of the *individual* (the *genotype*), rather than on the individual itself (the *phenotype*). Reproduction is the process whereby new individuals are introduced into a *population*. During sexual reproduction *recombination* (or *crossover*) occurs, transmitting to the offspring chromosomes that are a melange of both parents' genetic information. Mutation introduces small changes into the inherited chromosomes; it often results from copying errors during reproduction. Selection is a process guided by the Darwinian principle of survival of the fittest. The fittest individuals are those best adapted to their environment, which thus survive and reproduce.

Evolutionary computation makes use of a metaphor of natural evolution. According to this metaphor, a problem plays the role of an environment wherein lives a population of individuals, each representing a possible solution to the problem. The degree of adaptation of each individual (i.e. candidate solution) to its environment is expressed by an adequacy measure known as the *fitness function*. The phenotype of each individual, i.e. the candidate solution itself, is generally encoded in some manner into its *genome* (genotype). Like evolution in nature, evolutionary algorithms potentially produce progressively better solutions to the problem. This is possible thanks to the constant introduction of new 'genetic' material into the population, by applying so-called genetic operators that are the computational equivalents of natural evolutionary mechanisms.

There are several types of evolutionary algorithms, among which the best known are *genetic algorithms*, *genetic programming*, *evolution strategies*, and *evolutionary programming*; though different in the specifics they are all based on the same general principles. The archetypal evolutionary algorithm proceeds as follows: An initial population of individuals, $P(0)$, is generated at random or heuristically. Every evolutionary step $t$, known as a *generation*, the individuals in the current population, $P(t)$, are *decoded* and *evaluated* according to some predefined quality criterion, referred to as the fitness, or fitness function. Then, a subset of individuals, $P'(t)$ — known as the *mating pool* — is selected to reproduce, with selection of

individuals done according to their fitness. Thus, high-fitness ('good') individuals stand a better chance of 'reproducing', while low-fitness ones are more likely to disappear.

Selection alone cannot introduce any new individuals into the population, i.e. it cannot find new points in the search space. These points are generated by altering the selected population $P'(t)$ via the application of crossover and mutation, so as to produce a new population, $P''(t)$. Crossover tends to enable the evolutionary process to move toward 'promising' regions of the search space. Mutation is introduced to prevent premature convergence to local optima, by randomly sampling new points in the search space. Finally, the new individuals $P''(t)$ are introduced into the next-generation population, $P(t+1)$; usually $P''(t)$ simply becomes $P(t+1)$. The termination condition may be specified as some fixed, maximal number of generations or as the attainment of an acceptable fitness level. Fig. 1 presents the structure of a generic evolutionary algorithm in pseudo-code format.

As they combine elements of directed and stochastic search evolutionary techniques exhibit a number of advantages over other search methods. First, they usually need a smaller amount of knowledge and fewer assumptions about the characteristics of the search space. Second, they can more easily avoid getting stuck in local optima. Finally, they strike a good balance between *exploitation* of the best solutions, and *exploration* of the search space. The strength of evolutionary algorithms relies on their population-based search, and on the use of the genetic mechanisms described above. The existence of a population of candidate solutions entails a parallel search, with the selection mechanism directing the search to the most promising regions. The crossover operator encourages the exchange of information between these search-space regions, while the mutation operator enables the exploration of new directions.

The application of an evolutionary algorithm involves a number of important considerations. The first decision to take when applying such an algorithm is how to encode candidate solutions within the genome. The representation must allow for the encoding of all possible solutions while being sufficiently simple to be searched in a reasonable amount of time. Next, an appropriate fitness function must be defined for evaluating the individuals. The (usually scalar) fitness must reflect the

```
begin EC
    t:=0
    Initialize population P(t)
    while not done do
        Evaluate P(t)
        P'(t) := Select[P(t)]
        P''(t) := ApplyGeneticOperators[P'(t)]
        P(t+1) := Introduce[P''(t),P(t)]
        t:=t+1
    end while
end EC
```

Fig. 1. Pseudo-code of an evolutionary algorithm.

criteria to be optimized and their relative importance. Representation and fitness are thus clearly problem-dependent, in contrast to selection, crossover, and mutation, which seem *prima facie* more problem-independent. Practice has shown, however, that while standard genetic operators can be used, one often needs to tailor these to the problem as well.

We noted above that there are several types of evolutionary algorithms. This classification is due mainly to historical reasons and the different types of evolutionary algorithms are in fact quite similar. One could argue that there is but a single general evolutionary algorithm, or just the opposite — that "there are as many evolutionary algorithms as the researchers working in evolutionary computation" [78]. The frontiers among the widely accepted classes of evolutionary algorithms have become fuzzy over the years as each technique has attempted to overcome its limitations, by imbibing characteristics of the other techniques. To design an evolutionary algorithm one must define a number of important parameters, which are precisely those that demarcate the different evolutionary-computation classes. Some important parameters are: representation (genome), selection mechanism, crossover, mutation, size of populations $P'$ and $P''$, variability or fixity of population size, and variability or fixity of genome length.

The next subsections present the major evolutionary methods, emphasizing the specific properties of each one along with the most typical choices of parameters. These methods are: genetic algorithms (Section 2.1), genetic programming (Section 2.2), evolution strategies (Section 2.3), and evolutionary programming (Section 2.4). Section 2.5 then introduces a somewhat different evolutionary technique, known as *classifier systems*, which offers on-line learning capabilities. Finally, Section 2.6 outlines the principles for integrating evolutionary computation with other techniques, especially with *fuzzy logic* and *neural networks*. Detailed discussion on theory and on advanced topics of evolutionary computation can be found in [4,66,68,92,95].

## 2.1. Genetic algorithms

Proposed by John Holland in the 1960s [44,46], genetic algorithms are the best known class of evolutionary algorithms. They are used so extensively that often the terms genetic algorithms and evolutionary computation are used interchangeably (though, as noted, they should be considered distinct).

There is a clear distinction between the solution being tested, the 'adult individual' or phenotype, and its representation — the genome or genotype. Traditionally, the genome is a fixed-length binary string. With such a data structure it is possible to represent solutions to virtually any problem. However, so that the genetic algorithm may converge to good solutions, the representation must be carefully designed to minimize redundancy (i.e. several genotypes encoding the same phenotype) and to avoid invalid representations (i.e. a genotype encoding a phenotype which is not a possible solution to the problem at hand).

With genetic algorithms the population size is constant and individuals are decoded and evaluated at each generation. Individuals are then selected according

to their fitness. Many selection procedures are currently in use, one of the simplest being *fitness-proportionate* selection, where individuals are selected with a probability proportional to their fitness. This ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Selection produces a mating pool of the same size as the original population ($\|P'\| = \|P\|$).

Crossover is performed with probability $P_c$ (the 'crossover probability' or 'crossover rate') between two selected individuals, called parents, by exchanging parts of their genomes to form two new individuals, called offspring. In its simplest form, known as *one-point crossover*, substrings are exchanged after a randomly selected crossover point. The mutation operator is carried out by flipping bits at random, with some (usually small) probability $P_m$. The crossover and mutation operators preserve the size of the population, i.e. $\|P''\| = \|P'\|$.

Genetic algorithms are by far the most popular evolutionary technique (though genetic programming is rapidly 'gaining' on them). This is due in part to their conceptual simplicity, the ease of programming entailed, and the small number of parameters to be defined (apart from the genomic representation and the fitness function, parameters include mainly population size, crossover and mutation probabilities, and termination condition).

There are several variations of the simple genetic algorithm [95], with different selection mechanisms (e.g. ranking, tournament, and elitism), crossover operators (e.g. multi-point crossover), and mutation operators (e.g. adaptive mutation). These and other advanced topics concerning genetic algorithms are presented in Refs. [66,68,95].

## 2.2. Genetic programming

John Koza [50,51] developed relatively recently a variant of genetic algorithms called genetic programming. In this approach, instead of encoding possible solutions to a problem as a fixed-length character string, they are encoded as computer programs. To wit, the individuals in the population are programs that — when executed — are the candidate solutions (phenotypes) to the problem.

Programs in genetic programming may be expressed in any language in principle. However, to guarantee that evolution be able to generate valid, executable programs, it is necessary to restrict the choice of language. Thus, programs are expressed as parse trees, rather than as lines of code, i.e. using a functional language rather than a procedural one. The set of possible internal (non-terminal) nodes of these parse trees, called the *function set* ($F$), is composed of user-defined functions. The terminal nodes, which form the *terminal set* ($T$), are usually either variables or constants. The syntactic closure property requires that each function in $F$ be able to accept as arguments any other returned function value and any value and data type in the terminal set $T$. This property prevents the proliferation of illegal programs due to crossover and mutation.

As an example consider a basic arithmetic language whose function and terminal set are defined as follows: $F = \{ +, -, *, / \}$ and $T = \{A, B, C, 2\}$. Fig. 2 shows two examples of parse trees.
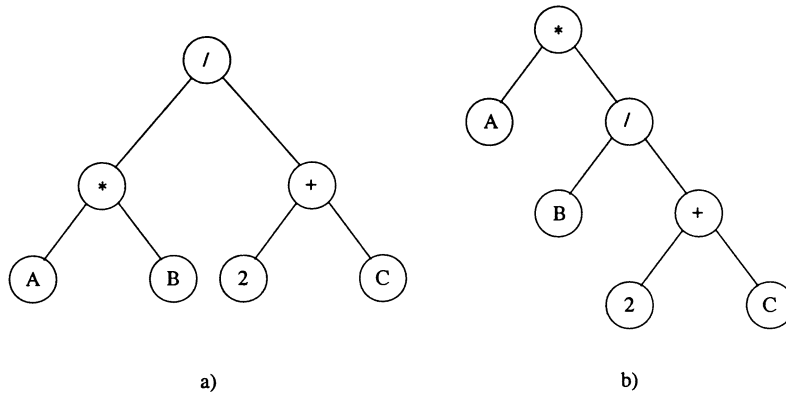
a)　　　　　　　　　　b)

Fig. 2. Genetic programming parse trees, representing the following programs in LISP-like syntax: (a) $(/(*AB)(+2C))$, and (b) $(*A(/B(+2C)))$. Both programs implement the expression $AB/(2+C)$. It is important to note that though LISP is the language chosen by Koza to implement genetic programming, it is not the only possibility. Any language capable of representing programs as parse trees is adequate. Moreover, machine language has been used as well [73].
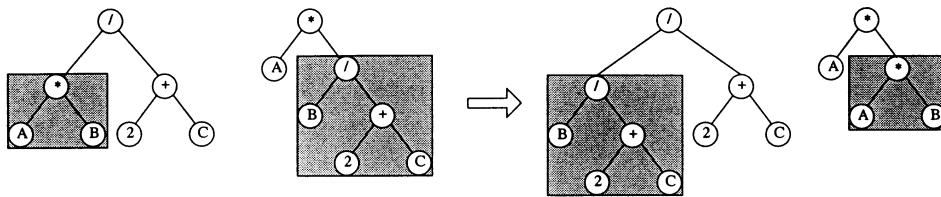


Fig. 3. Crossover in genetic programming. The two shadowed subtrees of the parent trees are exchanged to produce two offspring trees. Note that the two parents, as well as the two offspring, are typically of different size.

Evolution in genetic programming proceeds along the general lines of the generic evolutionary algorithm (Fig. 1), with the genetic operators adapted to the tree representation. Reproduction is performed in both asexual and sexual ways. Asexual reproduction, or cloning, allows some of the fittest individuals to survive into the next generation; this is equivalent to so-called elitist selection in genetic algorithms. Sexual reproduction, i.e. crossover, starts out by selecting a random crossover point in each parent tree and then exchanging the subtrees 'hanging' from these points, thus producing two offspring trees (Fig. 3). Mutation in genetic programming is considered as a secondary genetic operator and is applied much less frequently [51]. It is performed by removing a subtree at a randomly selected point and inserting at that point a new random subtree.

One important issue in genetic programming is related to the size of the trees. Under the influence of the crossover operator, the depth of the trees can quickly increase, leading to a fitness plateau. The presence of huge programs in the population also has direct consequences vis-a-vis computer memory and evaluation speed. Most implementations of genetic programming include mechanisms to

prevent trees from becoming too deep. However, these mechanisms also present a disadvantage, in that they reduce the genetic diversity contained in larger trees.

Further information on advanced topics and applications of genetic programming can be found in Ref. [6].

## 2.3. Evolution strategies

Evolution strategies were introduced by Ingo Rechenberg [81,82] and Hans Paul Schweffel [86,87] in the 1960s as a method for solving parameter-optimization problems. In its most general form the phenotype of an individual is a vector $\vec{x}$ containing the candidate values of the parameters being optimized. The genotype of each individual is a pair of real-valued vectors $\vec{v} = \{\vec{x}, \vec{\sigma}\}$, where $\vec{x}$ is the above phenotypic vector (the genotype–phenotype distinction is thus somewhat degenerate with evolution strategies), and $\vec{\sigma}$ is a vector of standard deviations (S.D.s) used to apply the mutation operator. The inclusion of the $\vec{\sigma}$ vector in the genome allows the algorithm to self-adapt the mutation operator while searching for the solution.

Somewhat different to the generic evolutionary algorithm (Fig. 1), selection is performed after the genetic operators have been applied. The standard notations in this domain, $(\mu, \lambda) - \text{ES}$ and $(\mu + \lambda) - \text{ES}$, denote algorithms in which a population of $\mu$ parents generates $\lambda$ offspring. The next generation is created by selecting the fittest $\mu$ individuals. In the case of $(\mu, \lambda) - \text{ES}$ only the $\lambda$ offspring are considered for selection, thus limiting the 'life' of an individual to one generation, while in the $(\mu + \lambda) - \text{ES}$ the $\mu$ parents are also considered for selection.

Mutation is the major genetic operator in evolution strategies. It also plays the role of a reproduction operator given that the mutated individual is viewed as an offspring for the selection operator to work on. In its most general form, mutation modifies a genotype $\vec{v} = \{\vec{x}, \vec{\sigma}\}$, by first randomly altering $\vec{\sigma}$, and then modifying $\vec{x}$ according to the new values provided by $\vec{\sigma}$. This operation produces a new individual $\vec{v}' = \{\vec{x}', \vec{\sigma}'\}$, where $\vec{x}' = \vec{x} + N(0, \vec{\sigma}')$. $N(0, \vec{\sigma}')$ denotes a vector of independent random Gaussian values with mean 0 and S.D.s $\vec{\sigma}'$.

The crossover (or recombination) operator generates an offspring from a number of parents (usually two). There are two types of crossover operators: discrete and intermediate. In discrete recombination each component of $\vec{v}$, i.e. each pair of scalars $(\vec{x}_i, \vec{\sigma}_i)$, is copied from one of the parents at random. In intermediate recombination, the offspring values are a linear combination of all the parent vectors participating in the recombination process.

The earliest evolution strategies were $(1 + 1) - \text{ES}$ [81,86], involving a single parent–single offspring search. Mutation was the only genetic operator. and the standard deviation vector $\vec{\sigma}$ was constant or modified by some deterministic algorithm. Later, recombination was added as evolution strategies were extended to encompass populations of individuals.

A good source for further information on evolution strategies is Ref. [88].

## 2.4. Evolutionary programming

Fogel [31,32] proposed evolutionary programming as a means to develop artificial intelligence. He argued that intelligent behavior requires both the ability to predict changes in an environment, and a translation of the predictions into actions appropriate for reaching a goal. In its most general, the environment is described as a sequence of symbols taken from a finite alphabet. With its knowledge of the environment the evolving entity is supposed to produce an output symbol that is related in some way to the next symbol appearing in the environment. The output symbol should maximize a *payoff function*, which measures the accuracy of the prediction. *Finite state machines* were selected to represent individuals in evolutionary programming as they provide a meaningful representation of behavior based on interpretation of symbols.

A finite state machine is a machine possessing a finite number of different internal states. When stimulated by an input symbol the machine undergoes a transition (i.e. a change in the internal state) and produces an output symbol. The behavior of the finite state machine is described completely by defining the (*output symbol*, *next state*) pair for each (*input symbol*, *current state*) pair. Fig. 4 shows an example of a three-state machine.

Evolutionary programming maintains a population of finite state machines, each one representing a particular candidate behavior for solving the problem at hand. The fitness of an individual is calculated by presenting sequentially to the finite state machine the symbols in the environment and observing the predicted output. The quality of the prediction is quantified according to the given payoff function. Once the individual has been exposed to the whole sequence of symbols, its overall performance (e.g. average payoff per symbol) is used as the fitness value.

Like with evolution strategies, evolutionary programming first generates offspring and then selects the next generation. There is no sexual reproduction (crossover), but rather each parent machine is mutated to produce a single offspring. There are five possible mutation operators: change of an output symbol,
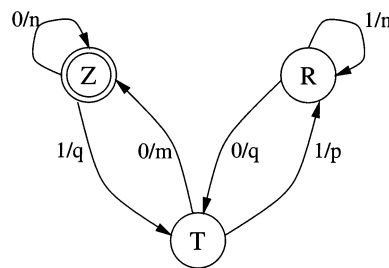


Fig. 4. A finite state machine with states {Z, T, R}. The input symbols belong to the set {0, 1}, and the output alphabet is the set {m, n, p, q}. The edges representing the state transitions are labeled a/b, where a represents the input symbol triggering the transition, and b represents the output symbol. For example, when the machine is in state R and the input is 0 it switches to state T and outputs q. A double circle indicates the initial state.

change of a state transition, addition of a state, deletion of a state, and change of the initial state. The two latter operations are not allowed when the parent machine has only one state. Mutation operators and the number of mutations per offspring are chosen with respect to a probability distribution. The offspring are then evaluated in the same way as their parents, and the next generation is selected from the ensemble of parents and offspring. This process is iterated until a new symbol is required in the environment. The best individual obtained up to this moment provides the prediction, the new symbol is added to the environment, and the algorithm is executed again. Note that as opposed to most evolutionary-computation applications where fitness is fixed from the outset, evolutionary programming inherently incorporates a dynamic fitness, i.e. the environment changes in time.

Fogel's book [29] is a good reference on evolutionary programming.

### 2.5. Classifier systems

Classifier systems, presented by Holland [45,46], are evolution-based learning systems, rather than a 'pure' evolutionary algorithm. They can be thought of as restricted versions of classical rule-based systems, with the addition of input and output interfaces. A classifier system consists of three main components: (1) the *rule and message system*, which performs the inference and defines the behavior of the whole system, (2) the *apportionment of credit system*, which adapts the behavior by credit assignment, and (3) the genetic algorithm, which adapts the system's knowledge by rule discovery.

The rule and message system includes four subcomponents: the *message list*, the *classifier list*, the *input interface*, and the *output interface*. The input interface, also known as the detector, translates information from the system's environment into one or more finite-length binary *messages*, which are posted to the finite-length message list. These messages may then activate one or more matching classifiers from the classifier list. A classifier is a rule of the form *if condition then message*, where *condition* is a finite-length string, and the *message*, posted to the message list, may then activate other classifiers or trigger a system's action through the output interface, also called the effector (the alphabet of classifiers includes the symbol # that plays the role of a wild-card character).

The apportionment-of-credit algorithm adapts the behavior of the classifier system by modifying the way existing classifiers are used. Unlike traditional rule-based systems, classifier systems use parallel rule activation. This characteristic allows the system to accelerate the inference process and to coordinate several actions simultaneously. However, with such a competitive approach the system must determine the importance (strength) of rules in order to combine them to make an overall decision. Although there are several ways to accomplish this, the bucket-brigade algorithm continues to be the most popular [35]. It is a parallel, local, credit-assignment-based reinforcement learning algorithm, which may be viewed as an 'information market', where the right to trade information is bought and sold by classifiers. Each matched classifier makes a bid proportional to its strength. Rules that have accumulated a large 'capital' (i.e. strength) are preferred over other rules.

The genetic algorithm adapts the classifier system by introducing new classifiers (rules). There exist two approaches for the application of evolutionary techniques in the design of rule-based systems in general: the Michigan approach and the Pittsburgh approach; these two approaches are also applied to classifier systems. In the Michigan approach each individual represents a single rule, and the classifier list is represented by the entire population. The strengths calculated by the bucket-brigade algorithm are used as a fitness function to evaluate the quality of each classifier. In the Pittsburgh approach the genetic algorithm maintains a population of candidate classifier lists, with each individual representing an entire list.

A good introduction to classifier systems is given by Goldberg [35].

## 2.6. Hybrid approaches: cooperating to succeed

There exist several so-called 'intelligent' techniques applicable to medical problems, evolutionary computation being one of them (and our focus herein). All the techniques have different capabilities and different limitations which determine their efficacy for a given problem. Hybrid approaches try to combine the properties of two or more intelligent techniques so as to enhance their capabilities and overcome their limitations. Herein, we briefly present some general considerations regarding the integration process that leads to hybrid systems, and then present two specific hybrid approaches: evolutionary-fuzzy and evolutionary-neural.

### 2.6.1. Integration of intelligent techniques

The integration of distinct techniques can be classified according to the amount of interactivity among the component approaches. We can distinguish between four levels of interactivity: uncoupled, loose coupling, tight coupling, and full integration [63,64].

In uncoupled systems, all the techniques are applied to the same problem but they do not interact with each other. In the simplest approach, the problem is decomposed into subproblems, and each technique solves the subproblem for which it is best suited. Another approach, called stand-alone, consists in the simultaneous application of the selected techniques to solve the same problem. A heuristic algorithm compares the independent solutions to then take the final decision. Systems which are loosely coupled also decompose the problem, but in this case the different techniques solving the subproblems communicate via data files. The main difference between loosely coupled systems and tightly coupled ones is that these latter communicate via memory resident data structures rather than via external data files, thus generally increasing overall performance. Finally, fully integrated hybrid systems share data structures and knowledge representation among the different search components. The communication between these components is implicit in the structures they share, and in the way these structures are designed. Below we briefly touch upon two types of hybrid systems, emphasizing full-integration aspects.

## 2.6.2. Evolutionary-fuzzy hybrid systems

Fuzzy logic is a computational paradigm that provides a mathematical tool for dealing with the uncertainty and the imprecision typical of human reasoning [97]. A prime characteristic of fuzzy logic is its capability of expressing knowledge in a linguistic way, allowing a system to be described by simple, 'human-friendly' rules. This characteristic, also known as interpretability, renders fuzzy logic-based systems attractive from the medical point of view.

A fuzzy inference system is a rule-based system that uses fuzzy logic, rather than Boolean logic, to reason about data [97]. Its basic structure comprises four main components: (1) a fuzzifier, which translates crisp (real-valued) inputs into fuzzy values, (2) an inference engine that applies a fuzzy reasoning mechanism to obtain a fuzzy output, (3) a defuzzifier, which translates this latter into a crisp value, and (4) a knowledge base, which contains both an ensemble of fuzzy rules, known as the rule base, and a database, which defines the membership functions used in fuzzy logic [75].

The design of a fuzzy inference system, i.e. the definition of its parameters so as to attain a desired behavior, is a hard task because of the quantity and the diversity of these parameters. This task can be considered as an optimization process where part or all of the parameters of a fuzzy system constitute the search space. This is where evolutionary algorithms step in, enabling the automatic design of fuzzy systems, based on a database of training cases.

Depending on several criteria, evolutionary computation can be applied in different stages of the fuzzy-parameter search. In the simplest approach, an evolutionary algorithm is used to find the adequate membership function values contained in the database. The design of the rule base, and of the entire knowledge base, can be viewed as rule-based learning processes with different levels of complexity. They can thus be assimilated within other methods of machine learning, taking advantage of experience gained in this latter domain. The aforementioned Michigan and Pittsburgh approaches (see Section 2.5) are widely applied in evolutionary-fuzzy modeling. A more recent method proposed specifically for fuzzy systems is the iterative rule learning approach [42].

## 2.6.3. Evolutionary-neural hybrid systems

Evolution and learning [3] are two forms of adaptation, the former adapting populations by globally optimizing their collective performance, while the latter adapts individuals by locally optimizing their individual performance. Evolutionary artificial neural networks refers to a special class of artificial neural networks where evolution is combined with learning to attain adaptation. In the simplest approach, evolutionary algorithms search for network connection weights, thus substituting the learning rule. More sophisticated approaches use evolutionary techniques to perform various other tasks, such as topology definition, learning-rule adaptation, input feature selection, and connection-weights initialization [99].

## 3. Applying evolutionary computation to solve medical problems

Most (if not all) medical decisions can be formulated as a search in some appropriate space. For example. a pathologist analyzing biopsies to decide whether they are malignant or not, is searching in the space of all possible cell features for a set of features permitting him to provide a clear diagnosis [75]. A radiologist planning a sequence of radiation doses is searching for the best treatment in the space of all possible treatments [103].

Medical search spaces are usually very large and complex. Medical decisions are based on clinical tests which provide huge amounts of data. Based on these data one must ultimately make a single decision (e.g. benign or malignant). Given the tight interdependency among the domain variables, and the inherent non-linearity of most real-world problems, neighboring points in the search space may have widely differing qualities, turning the search into a complex task. Indeed, due to this complexity, several medical problems are used as benchmarks to test and compare machine learning techniques [65,67].

Evolutionary computation provides powerful techniques for searching complex spaces. As stated in Section 2 evolutionary techniques exploit mechanisms of natural evolution to search efficiently in a given space. Their intrinsic parallelism diminishes the risk of the search being trapped in a local optimum.

The construction of accurate models of medical decision from extant knowledge is a hard task. On the one hand, the models involve too many non-linear and uncertain parameters to be treated analytically. On the other hand, medical experts are usually not available, or simply do not collaborate in translating their experience into a usable decision tool.

Nevertheless, there is a large number of accessible medical databases. Currently, medical results are electronically stored and accumulated in databases so as to serve both as a record of patients' history, and as a source of medical knowledge. The amount of available data is increasing continuously and, therefore, its exploitation requires the use of more sophisticated computational processing tools.

Evolutionary computation is applied in medicine to perform several types of tasks. Whenever a decision is required in medicine, it is usually possible to find a niche for evolutionary techniques. The tasks performed by evolutionary algorithms in the medical domain can be divided into three groups: (1) data mining, mainly as applied to diagnosis and prognosis, (2) medical imaging and signal processing, and (3) planning and scheduling.

### 3.1. Data mining

Data mining, also known as knowledge discovery, is the process of finding patterns, trends, and regularities by sifting through large amounts of data [26]. Data mining involves the analysis of data stored in databases to discover associations or patterns, to segment (or cluster) records based on similarity of attributes, and to create predictive (or classification) models.

There are two major approaches to data mining: supervised and unsupervised. In the supervised approach, specific examples of a target concept are given, and the goal is to learn how to recognize members of the class using the description attributes. In the unsupervised approach, a set of examples is provided without any prior classification, and the goal is to discover underlying regularities and patterns, most often by identifying clusters or subsets of similar examples.

Clinical databases have accumulated large amounts of data on patients and their medical conditions. The clinical history of a patient generates data that goes beyond the disease being treated. This information, stored along with that of other patients, constitutes a good place to look for new relationships and patterns, or to validate proposed hypotheses.

The range of applications of data mining in medicine is very wide, with the two most popular applications being diagnosis and prognosis. Diagnosis is the process of selectively gathering information concerning a patient, and interpreting it according to previous knowledge, as evidence for or against the presence or absence of disorders [58]. In a prognostic process, a patient's information is also gathered and interpreted, but the objective is to predict the future development of the patient's condition. Due to the predictive nature of this process, prognostic systems are frequently used as tools to plan medical treatments [59].

The role played by data mining in the context of diagnosis and prognosis is the discovery of the knowledge necessary to interpret the gathered information. In some cases this knowledge is expressed as probabilistic relationships between clinical features and the proposed diagnosis or prognosis. In other cases a rule-based representation is chosen so as to provide the physician with an explanation of the decision. Finally, in yet other cases, the system is designed as a black-box decision maker that is totally unconcerned with the interpretation of its decisions.

Evolutionary computation is usually applied in medical data mining as a parameter finder. Evolutionary techniques search for the parameter values of the knowledge representation set up by the designer so that the mined data are optimally interpreted. For example, evolutionary algorithms can search for the weights of a neural network, the membership function values of a fuzzy system, or the coefficients of a linear regressor.

## 3.2. Medical imaging and signal processing

Much of the medical data are expressed as images or other types of temporal signals. Many exams, such as magnetic resonance or mammography, provide as a result a group of images. Sometimes the number of images is high and the important information is distributed among all of them. Other types of exams, like electroencephalography (EEG), provide results in the form of time-variant signals. In these tests the information is hidden within the temporal features. The fields of signal and image processing have developed tools to deal with such

data. Techniques such as filtering, compression, segmentation, and pattern recognition allow for the extraction of the desired features from the signals.

Signal processing in medicine is subject to several important constraints. First, the number of signals to be processed is high, and often tightly interdependent. Second, signals are unique, in the sense that the circumstances under which they were obtained are normally not repeatable. Third, given the characteristics of their sources, medical signals are often very noisy. Finally, in some cases information about the signals is required in real time in order to take crucial decisions.

Evolutionary-computation techniques are used in different ways in medical imaging and signal processing. In some cases they are applied to improve the performance of signal-processing algorithms (e.g. filters or compressors) by finding their optimal parameters. Other works use evolutionary algorithms directly to extract useful information from the welter of data.

### 3.3. Planning and scheduling

Evolutionary computation is frequently applied in problems of planning and scheduling. These tasks involve the assignment of resources subject to several constraints. In a planning problem several resources (or operations) are needed in order to accomplish a task, and the planning process tries to define the sequence in which they are applied. In a scheduling problem one or several resources have to be distributed among several tasks, and the problem is to find a distribution (spatial and temporal) for this assignment so that all tasks are executed. Evolutionary techniques are well suited to solve these kinds of problems thanks to their ability to search complex spaces.

In medicine planning and scheduling are widely used in clinical and hospital administration applications. Clinical applications involve mainly planning problems; e.g. in 3D radiotherapy it is necessary to plan carefully the doses applied for several different radioactive beams, respecting clinical and geometrical constraints [25]. Hospital administrators deal with problems of personnel, patient, and resource management and control. An example is the problem of scheduling a patient for undergoing different medical procedures and seeing different physicians, optimizing both patient waiting time and apparatus utilization [77].

## 4. Classified bibliography

This section contains a bibliography of articles dealing with evolutionary computation in medicine, classified both according to the medical task and according to the evolutionary technique. We have chosen to concentrate mainly on articles in archival journals so as to limit the explosive number of references. Table 1 summarizes these two classifications.

Table 1
Classified bibliography: summary

| Evolutionary technique | Medical task | | | |
|---|---|---|---|---|
| | Data mining | | Medical imaging and signal processing | Planning and scheduling |
| | Diagnosis | Prognosis | | |
| *Genetic algorithms* | | | | |
| Unidimensional, binary | [7–9,47,54,57,75,79,83] | [1,16,19,24,48,49,54,55, 60,61,70,89,90] | [5,15,18,21,23,27,37,41,43,52,69,72,84,91, 93,94,96,105] | [98,101–103] |
| Multidimensional | | | [20,22,39] | [17,77,104] |
| Real-valued | [38] | | [2,62,85] | [25,40,56] |
| Rule-encoding | [10,11] | | | [11] |
| Indexed | | | [53] | |
| Genetic programming | [10,36,71] | | | |
| Evolution strategies | [7–9] | | | [76] |
| Evolutionary programming | [30,71,100] | | | |
| Classifier systems | [12,74,80] | [12] | | |
| *Hybrid systems* | | | | |
| Evolutionary-fuzzy | [7–9,47,75] | [55] | [94] | |
| Evolutionary-neural | [7–9,30,74,100] | [16,19,24,48,49,60,70,90] | [18,23,41] | |

## 4.1. According to the medical task

### 4.1.1. Data mining

*4.1.1.1. Diagnosis.* Diagnosis is the process of selectively gathering information concerning a patient, and interpreting it according to previous knowledge, as evidence for or against the presence or absence of disorders (Section 3.1) and refs [7–12,30,36,38,47,54,57,71,74,75,79,80,83,100]. The papers in this category apply evolutionary algorithms to solve numerous diagnostic problems, including: the patient's general condition evaluation, location of primary tumor, detection of hyaline membrane disease in preterm newborn infants, detection of breast cancer cells in peripheral blood flow, lymphography classification, brain tumor classification, risk evaluation of heart and coronary artery diseases, profiling of risk factors in diabetes and female urinary incontinence, and the detection of different diseases from clinical tests (infarcts from radiography, arrythmias from electrocardiography, neuromuscular disorders from electromyography, sleep profiling from electroencephalography, and breast cancer from mammography).

*4.1.1.2. Prognosis.* Prognosis is the process of selectively gathering information concerning a patient, and interpreting it according to previous knowledge, in order to predict the future development of the patient's condition (Section 3.1) and refs [1,12,16,19,24,48,49,54,55,60,61,70,89,90]. The papers in this category involve prognostic applications of evolutionary computation, including: breast cancer recurrency, donor compatibility for transplants in highly sensitized patients, outcome of duodenal ulcer, hypoxic resistance on jet pilots, outcome of intensive-care patients, after-surgery response for patients with lung cancer, prediction of depression after mania, prevision of tractolimus blood level in liver transplantation patients, and patient's survival estimation in different types of cancer (malignant skin melanoma, lung cancer, colorectal cancer, and gestational trophoblastic tumours).

### 4.1.2. Medical imaging and signal processing

The fields of medical imaging and signal processing have developed tools to deal with huge amounts of data expressed as images or other types of temporal signals [2,5,15,18,20–23,27,37,39,41,43,52,53,62,69,72,84,85,91,93,94,96,105]. All but one of the papers in this category deal with problems related to clinical tests, including: thorax radiography, retinal and cardiac angiography, computarized tomography, magnetoencephalography, ultrasound imaging, electroencephalography, electrocardiography, radiographic cephalography, laser profilometry, and many applications of both mammography and magnetic resonance. The 'odd paper out' presents an application of surgery assistance [5].

### 4.1.3. Planning and scheduling

Planning and scheduling involve the assignment of resources to accomplish one or more tasks subject to several constraints (Section 3.3) and refs [11,17,25,40,56,76,77,98,101–104]. The papers in this category use evolutionary

techniques to solve problems such as: allocation of hospital resources, electrical carotid sinus nerve stimulation, radiologist allocation, three-dimensional radiation therapy treatment planning, dosimetric preplanning and treatment planning of permanent prostate implants, patient scheduling in highly constrained situations, and stereotactic radiosurgery planning.

## 4.2. According to the evolutionary technique

### 4.2.1. Genetic algorithms

As mentioned in Section 2 genetic algorithms are the best-known class of evolutionary algorithms and their use is so extensive that often the terms genetic algorithms and evolutionary computation are used interchangeably. The main difference between genetic algorithms and other evolutionary algorithms is the representation (genome), which we use as a criterion to further subdivide this class:

*4.2.1.1. Unidimensional, binary genome.* This is the most popular representation in genetic algorithms [1,5,7–9,15,16,18,19,21,23–27,37,41,43,47–49,52,54,55,57,60,61,-69,70,72,75,79,83,84,89–91,93,94,96,98,101–103,105] because it is simple to use and applicable to almost any problem [92].

*4.2.1.2. Multidimensional genome.* In many cases the nature of the problem (e.g. bidimensional medical imaging problems) suggests the use of matrices or multidimensional arrays to represent the candidate solutions [17,20,22,39,77,104].

*4.2.1.3. Real-valued genome.* The binary representation has some drawbacks when applied to high-precision numerical problems [66]. In particular, for parameter-optimization problems with variables from continuous domains, a real-valued representation is sometimes more efficient (and more natural) than a binary one [2,25,38,40,56,62,85].

*4.2.1.4. Rule-encoding genome.* Rule-based systems are usually encoded in genetic algorithms using a binary representation. However, some works, such as the two papers of this subcategory, represent directly in the genome the rules of the knowledge base [10,11].

*4.2.1.5. Indexed representation.* In this representation the genome is encoded using a finite-length alphabet (non-binary) [53]. Each symbol of the alphabet indexes an element, and the presence of a symbol in the genotype indicates the presence of the element in the phenotype. In the referenced paper the indexed elements are 'leafs' of a known tree representation (note that this is still a genetic-algorithm string representation and not a genetic-programming tree).

### 4.2.2. Genetic programming

In genetic programming solutions are encoded as computer programs rather than as fixed-length character strings (Section 2.2) [10,36,71].

### 4.2.3. Evolution strategies

Evolution strategies are well suited for parameter-optimization problems [7–9,76]. They use mainly the mutation operator. A major characteristic of evolution strategies is that mutation values are evolved along with the parameters being optimized (Section 2.3).

### 4.2.4. Evolutionary programming

In evolutionary programming, individuals are represented by finite state machines, which provide a meaningful representation of behavior based on interpretation of environmental symbols (Section 2.4) [30,71,100].

### 4.2.5. Classifier systems

Classifier systems are evolution-based learning systems [12,74,80]. They can be viewed as restricted versions of classical rule-based systems that add interaction with the exterior thanks to input and output interfaces (Section 2.5).

### 4.2.6. Hybrid approaches

*4.2.6.1. Evolutionary-fuzzy systems.* In evolutionary-fuzzy systems the capability of expressing knowledge in a linguistic, 'human-friendly' way offered by fuzzy logic, is combined with the power of evolutionary algorithms to search and optimize [7–9,47,55,75,94]. Thus one obtains systems with both high performance as well as high interpretability (Section 2.6).

*4.2.6.2. Evolutionary-neural systems.* With evolutionary-neural systems evolution and learning strategies work in concert to attain adaptation (Section 2.6) [7–9,16,18,19,23,24,30,41,48,49,60,70,74,90,100].

## References

[1] Anand SS, Smith AE, Hamilton PW, Anand JS, Hughes JG, Bartels PH. An evaluation of intelligent prognostic systems for colorectal cancer. Artif Intell Med 1999;15(2):193–214.

[2] Anastasio MA, Yoshida H, Nagel R, Nishikawa RM, Doi K. A genetic algorithm-based method for optimizing the performance of a computer-aided diagnosis scheme for detection of clustered microcalcifications in mammograms. Med Phys 1998;25(9):1613–20.

[3] Arbib MA. Handbook of Brain Theory and Neural Networks. Cambridge, MA: MIT Press, 1995.

[4] Back T, Hammel U, Schwefel H-P. Evolutionary computation: comments on the history and current state. IEEE Trans Evol Comput 1997;1(1):3–17.

[5] Baluja S, Simon D. Evolution-based methods for selecting point data for object localization: applications to computer-assisted surgery. Appl Intell 1998;8(1):7–19.

[6] Banzhaf W, Nordin P, Keller RE, Francone FD. Genetic Programming — An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, 1998.

[7] Baumgart-Schmitt R, Eilers R, Herrmann WM. On the use of neural network techniques to analyse sleep EEG data. Second communication: training of evolutionary optimized neural networks on the basis of multiple subjects data and the application of context rules according to rechtschaffen and Kales. Somnologie 1997;1(4):171–83.

[8] Baumgart-Schmitt R, Herrmann WM, Eilers R. On the use of neural network techniques to analyze sleep EEG data. Third communication: robustification of the classificator by applying an algorithm obtained from 9 different networks. Neuropsychobiology 1998;37(1):49–58.

[9] Baumgart-Schmitt R, Herrmann WM, Eilers R, Bes F. On the use of neural network techniques to analyse sleep EEG data. First communication: application of evolutionary and genetic algorithms to reduce the feature space and to develop classification rules. Neuropsychobiology 1997;36(4):194–210.

[10] Bickel AS, Bickel RW. Tree structured rules in genetic algorithms. In: Grefenstette JJ, editor. Genetic Algorithms and their Applications: Proceedings of the second International Conference on Genetic Algorithms, MIT, Cambridge, MA. USA, July 1987. Lawrence Erlbaum Associates, pp. 77-81.

[11] Bickel AS, Bickel RW. Determination of near-optimum use of diagnostic resources using the 'GENES' genetic algorithm shell. Comput Biol Med 1990;20(1):1–13.

[12] Bonelli P, Parodi A. An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains. In: Belew R, Booker L, editors. Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, CA, July 1991. Morgan Kaufman, pp. 288-295.

[13] Box GEP. Evolutionary operation: a method for increasing industrial productivity. App Stat 1957;6(2):81–101.

[14] Box GEP, Hunter JS. Condensed calculations for evolutionary operation programs. Technometrics 1959;1:77–95.

[15] Chan H-P, Sahiner B, Lam KL, et al. Computerized analysis of mammographic microcalcifications in morphological and texture feature spaces. Med Phys 1998;25(10):2007–19.

[16] Chen H-Y, Chen TC, Min DI, Fischer GW, Wu Y-M. Prediction of tacrolimus blood levels by using the neural network with genetic algorithm in liver transplantation patients. Ther Drug Monit 1999;21(1):50–6.

[17] Chen Y, Narita M, Tsuji M, Sa S. A genetic algorithm approach to optimization for the radiological worker allocation problem. Health Phys 1996;70(2):180–6.

[18] Chen Y-T, Cheng K-S, Liu J-K. Improving cephalogram analysis through feature subimage extraction. IEEE Eng Med Biol Mag 1999;18(1):25–31.

[19] Clark BD, Leong SW. Crossmatch prediction of highly sensitized patients. Clin Transpl 1992;435-55.

[20] Delibasis KK, Undrill PE, Cameron G. Designing texture filters with genetic algorithms: an application to medical images. Signal Process 1997;57:19–33.

[21] Delibasis KK, Undrill PE, Cameron GG. Genetic algorithm implementation of stack filter design for image restoration. IKE Proc Vis Image Signal Process 1996;143(3):177–83.

[22] Delibasis KK, Undrill PE, Cameron GG. Designing Fourier descriptor-based geometric models for object interpretation in medical images using genetic algorithms. Comput Vis Image Underst 1997;66(3):286–300.

[23] Dhawan AP, Chitre Y, Kaiser-Bonasso C. Analysis of mammographic microcalcifications using gray-level image structure features. IEEE Trans Med Imaging 1996;15(3):246–59.

[24] Dybowski R, Weller P, Chang R, Gant V. Prediction of outcome in critically ill patients using artificial neural network synthesised by genetic algorithm. Lancet 1996;347(9009):1146–50.

[25] Ezzell GA. Genetic and geometric optimization of three-dimensional radiation therapy treatment planning. Med Phys 1996;23(3):293–305.

[26] Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R, editors. Advances in Knowledge Discovery and Data Mining. Cambridge, MA: AIII Press/MIT Press, 1996.

[27] Fitzpatrick JM, Grefenstette JJ, Van Gucht D. Image registration by genetic search. In: Proc. of the IEEE Southeast Conference SOUTHEASTCON 84, 1984:460-464.

[28] Fogel DB III, editor. Evolutionary Computation: The Fossil Record. Piscataway, NJ: IEEE Press, 1998.

[29] Fogel DB. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 2nd edn. Piscataway, NJ: IEEE Press, 1999.

[30] Fogel DB, Wasson EC III, Boughton EM, Porto VW. Evolving artificial neural networks for screening features from mammograms. Artif Intell Med 1998;14(3):317–26.

[31] Fogel LJ. Autonomous automata. Ind Res 1962;4:14–9.

[32] Fogel LJ, Owens AJ, Walsh MJ. Artificial Intelligence through Simulated Evolution. New York: Wiley, 1966.

[33] Friedberg RM. A learning machine: I. IBM J Res Dev 1958;2:2–13.

[34] Friedberg RM, Dunham B, North JH. A learning machine. II. IBM J Res Dev 1959;3:282–7.

[35] Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[36] Gray HF, Maxwell RJ, Martinez-Perez I, Arus C, Cerdan S. Genetic programming for classification and feature selection: analysis of $^1$H nuclear magnetic resonance spectra from human brain tumour biopsies. NMR Biomed 1998;11(4-5):217–24.

[37] Grefenstette JJ, Fitzpatrick MJ. Genetic search with approximate function evaluation. In: Grefenstette JJ, editor. Proc of the 1st International Conference on Genetic Algorithms and their Applications, Pittsburgh, PA, July 1985. Lawrence Erlbaum Associates, pp. 112–120.

[38] Gross HJ, Verwer B, Houck D, Hoffman RA, Recktenwald D. Model study detecting breast cancer cells in peripheral blood mononuclear cells at frequencies as low as $10(-7)$. Proc Natl Acad Sci USA 1995;92(2):537–41.

[39] Gudmundsson M, El-Kwae EA, Kabuka MR. Edge detection in medical images using a genetic algorithm. IEEE Trans Med Imaging 1998;17(3):469–74.

[40] Haas OCL, Burnham KJ, Mills JA. Optimization of beam orientation in radiotherapy using planar geometry. Phys Med Biol 1998;43(8):2179–93.

[41] Handels H, Rop T, Kreusch J, Wolff HH, Poppl SJ. Feature selection for optimized skin tumor recognition using genetic algorithms. Artif Intell Med 1999;16(3):283–97.

[42] Herrera F, Lozano M, Verdegay JL. Generating fuzzy rules from examples using genetic algorithms. In: Bouchon-Meunier B, Yager RR, Zadeh LA, editors. Fuzzy Logic and Soft Computing. Singapore: World Scientific, 1995:11–20.

[43] Hill A, Cootes TF, Taylor CJ, Lindley K. Medical image interpretation: a generic approach using deformable templates. Med Inform 1994;19(1):47–59.

[44] Holland JH. Outline for a logical theory of adaptive systems. J ACM 1962;9(3):297–314.

[45] Holland JH. Processing and processors for schemata. In: Jacks EL, editor. Associative Information Processing. New York: Elsevier, 1971:127–46.

[46] Holland JH. Adpatation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[47] Jain R, Mazumdar J, Moran W. Application of fuzzy-classifier system to coronary artery disease and breast cancer. Australas Phys Eng Sci Med 1998;21(3):141–7.

[48] Jefferson MF, Pendleton N, Lucas CP, Lucas SB, Horan MA. Evolution of artificial neural network architecture: prediction of depression after mania. Methods Inf Med 1998;37(3):220–5.

[49] Jefferson MF, Pendleton N, Lucas SB, Horan MA. Comparison of a genetic algorithm neural network with logistic regression for predicting outcome after surgery for patients with nonsmall cell lung carcinoma. Cancer 1997;79(7):1338–42.

[50] Koza, JR. Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STANCS-90-1314, Department of Computer Science, Stanford University, June 1990.

[51] Koza JR. Genetic Programming. Cambridge, MA: MIT Press, 1992.

[52] Kumaravel N, Nithlyanandam N. Genetic-algorithm cancellation of sinusoidal powerline interference in electrocardiograms. Med Biol Eng Comput 1998;36(2):191–6.

[53] Kumaravel N, Rajesh J, Nithlyanandam N. Equivalent tree representation of electrocardiogram using genetic algorithm. Biomed Sci Instrum 1997;33:573–8.

[54] Kuncheva L. Genetic algorithm for feature selection for parallel classifiers. Inf Process Lett 1993;46(4):163–8.

[55] Kuncheva L, Andreeva K. Dream: a shell-like software system for medical data analysis and decision support. Comput Methods Programs Biomed 1993;40(2):73–81.

[56] Langer M, Brown R, Morrill S, Lane R, Lee O. A generic genetic algorithm for generating beam weights. Med Phy 1996;23(6):965–71.

[57] Laurikkala J, Juhola M. A genetic-based machine learning system to discover the diagnostic rules for female urinary incontinence. Comput Methods Programs Biomed 1998;55(3):217–28.

[58] Lucas PJF. Analysis of notions of diagnosis. Artif Intell 1998;105(12):295–343.

[59] Lucas PJF. Prognostic methods in medicine. Artif Intell Med 1999;15:105–19.

[60] Marchevsky AM, Patel S, Wiley KJ, et al. Artificial neural networks and logistic regression as tools for prediction of survival in patients with stages I and II non-small cell lung cancer. Mod Pathol 1998;11(7):618–25.

[61] Marvin N, Bower M, Rowe JE. An evolutionary approach to constructing prognostic models. Artif Intell Med 1999;15(2):155–65.

[62] Matsopoulos GK, Mouravliansky NA, Delibasis KK, Nikita KS. Automatic retinal image registration scheme using global optimization techniques. IEEE Trans Inf Technol Biomed 1999;3(1):47–60.

[63] Medsker LR. Hybrid Neural Network and Expert System. Boston: Kluwer, 1994.

[64] Medsker LR, Bailey DL. Models and guidelines for integrating expert systems and neural networks. In: Kandel A, Langholz G, editors. Hybrid Architectures for Intelligent Systems. Boca Raton, FL: CRC Press, 1992:153–71.

[65] Merz CJ, Murphy PM. UCI repository of machine learning databases. http://www.ics.uci.edu/∼ mlearn/MLRepository.html, 1996.

[66] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Berlin: Springer, 1996.

[67] Michie D, Spiegelhalter DJ, Taylor CC, editors. Machine Learning, Neural and Statistical Classification. Chichester, UK: Ellis Horwood, 1994.

[68] Mitchell M. An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.

[69] Mitra SK, Sarbadhikari SN. Iterative function system and genetic algorithm-based EEG compression. Med Eng Phys 1997;19(7):605–17.

[70] Narayanan MN, Lucas SB. A genetic algorithm to improve a neural network to predict a patient's response to warfarin. Methods Inf Med 1993;32(1):55–8.

[71] Ngan PS, Wong ML, Lam W, Leung KS, Cheng JC. Medical data mining using evolutionary computation. Artif Intell Med 1999;16(1):73–96.

[72] Nikulin AE, Dolenko B, Bezabeh T, Somorjal RL. Near-optimal region selection for feature space reduction: novel preprocessing methods for classifying MR spectra. NMR Biomed 1998;11(4-5):209–16.

[73] Nordin P. Evolutionary Program Induction of Binary Machine Code and its Application. Munster, Germany: Krehl, 1997.

[74] Pattichis CS, Schizas CN. Genetics-based machine learning for the assessment of certain neuromuscular disorders. IEEE Trans Neural Netw 1996;7(2):427–39.

[75] Peña-Reyes CA, Sipper M. A fuzzy-genetic approach to breast cancer diagnosis. Artif Intell Med 1999;17(2):131–5.

[76] Peters TK, Koralewski HE, Zerbst EW. The evolution strategy — a search strategy used in individual optimization of electrical parameters for therapeutic carotid sinus nerve simulation. IEEE Trans Biomed Eng 1989;36(7):668–75.

[77] Podgorelec V, Kokol P. Genetic algorithm based system for patient scheduling in highly constrained situations. J Med Syst 1997;21(6):417–27.

[78] Poli R. Introduction to evolutionary computation. http://www.cs.bham.ac.uk/∼rmp/slide_book/, October 1996. visited: 16 March 1999.

[79] Poli R, Cagnoni S, Valli G. Genetic design of optimum linear and nonlinear QRS detectors. IEEE Trans Biomed Eng 1995;42(11):1137–41.

[80] Rada R, Rhine Y, Smailwood J. Rule refinement. In: Proc of the Society of Computer Applications in Medical Care, Vol. 8 of Annual Symposium on Computer Application in Medical Care; Proceedings, 1984:62-65.

[81] Rechenberg I. Cybernetic solution path of an experimental problem. Farborough Hants: Royal Aircraft Establishment. Library Translation 1122, August 1965. English Translation of lecture given at the Annual Conference of the WGLR at Berlin in September, 1964.

[82] Rechenberg I. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Frommann-Holzboog, 1973.

[83] Sahiner B, Chan HP, Petrick N, Helvie MA, Goodsitt MM. Design of a high-sensitivity classifier based on a genetic algorithm: application to computer-aided diagnosis. Phys Med Biol 1998;43(10):2853–71.

[84] Sahiner B, Chan HP, Wei D, et al. Image feature selection by a genetic algorithm: application to classification of mass and normal breast tissue. Med Phys 1996;23(10):167–84.

[85] Schroeter P, Vesin JM, Langenberger T, Meuli R. Robust parameter estimation of intensity distributions for brain magnetic resonance images. IEEE Trans Med Imaging 1998;17(2):172–86.

[86] Schwefel H-P. Kybernetische evolution als strategic der experimentelen forschung in der stromungstechnik. Master's Thesis, Technical University of Berlin, March 1965.

[87] Schwefel H-P. Evolutionsstrategie und numerische Optimierung. PhD Thesis, Technische Universitat Berlin, Berlin, May 1975.

[88] Schwefel H-P. Evolution and Optimum Seeking. Sixth-Generation Computer Technology Series. New York: Wiley, 1995.

[89] Sierra B, Larranaga P. Predicting survival in malignant skin melanoma using bayesian networks automatically induced by genetic algorithms. An empirical comparison between different approaches. Artif Intell Med 1998;14(1-2):215–30.

[90] Singson RP, Alsabeh R, Geller SA, Marchevsky A. Estimation of tumor stage and lymph node status in patients with colorectal adenocarcinoma using probabilistic neural networks and logistic regression. Mod Pathol 1999;12(5):479–84.

[91] Sonka M, Tadikonda SK, Collins SM. Knowledge-based interpretation of MR brain images. IEEE Trans Med Imaging 1996;15(4):443–52.

[92] Tettamanzi A, Tomassini M. Evolutionary algorithms and their applications. In: Mange D, Tomassini M, editors. Bio-Inspired Computing Machines: Toward Novel Computational Machines. Lausanne, Switzerland: Presses Polytechniques et Universitaires Romandes, 1998:59–98.

[93] Uutela K, Hamalainen M, Salmelin R. Global optimization in the localization of neuromagnetic sources. IEEE Trans Biomed Eng 1998;45(6):716–23.

[94] Velthuizen RP, Hall LO, Clarke LP. Feature extraction for MRI segmentation. J Neuroimaging 1999;9(2):85–90.

[95] Vose MD. The Simple Genetic Algorithm. Cambridge, MA: MIT Press, 1999.

[96] Wanschura T, Coley DA, Vennart W, Gandy S. Automatic realignment of time-separated MR images by genetic algorithm. Magn Reson Imaging 1999;17(2):313–7.

[97] Yager RR, Zadeh LA. Fuzzy Sets, Neural Networks, and Soft Computing. New York: Van Nostrand Reinhold, 1994.

[98] Yang G, Reinstein LE, Pai S, Xu Z, Carroll DL. A new genetic algorithm technique in optimization of permanent 1251 prostate implants. Med Phys 1998;25(12):2308–15.

[99] Yao X. Evolving artificial neural networks. Proc IEEE 1999;87(9):1423–47.

[100] Yao X, Liu Y. A new evolutionary system for evolving artificial neural networks. IEEE Trans Neural Netw 1997;8(3):694–713.

[101] Yu Y. Multiobjective decision theory for computational optimization in radiationtherapy. Med Phys 1997;24(9):1445–54.

[102] Yu Y, Schell MC. A genetic algorithm for the optimization of prostate implants. Med Phys 1996;23(12):2085–91.

[103] Yu Y, Schell MC, Zhang JB. Decision theoretic steering and genetic algorithm optimization: application to stereotactic radiosurgery treatment planning. Med Phys 1997;24(11):1742–50.

[104] Yu Y, Zhang JB, Brasacchio RA, et al. Automated treatment planning engine for prostate seed implant brachytherapy. Int J Radiat Oncol Biol Phys 1999;43(3):647–52.

[105] Zheng B, Chang YH, Wang XH, Good WF, Gur D. Feature selection for computerized mass detection in digitized mammograms by using a genetic algorithm. Acad Radiol 1999;6(6):327–32.