Engineering Evolutionary Intelligent Systems: Methodologies, Architectures and Reviews

Ajith Abraham and Crina Grosan

Summary. Designing intelligent paradigms using evolutionary algorithms is getting popular due to their capabilities in handling several real world problems involving complexity, noisy environment, imprecision, uncertainty and vagueness. In this Chapter, we illustrate the various possibilities for designing intelligent systems using evolutionary algorithms and also present some of the generic evolutionary design architectures that has evolved during the last couple of decades. We also provide a review of some of the recent interesting evolutionary intelligent system frameworks reported in the literature.

1 Introduction

Evolutionary Algorithms (EA) have recently received increased interest, particularly with regard to the manner in which they may be applied for practical problem solving. Usually grouped under the term evolutionary computation or evolutionary algorithms, we find the domains of Genetic Algorithms [34], Evolution Strategies [68], [69], Evolutionary Programming [20], Learning Classifier Systems [36], Genetic Programming [45], Differential Evolution [67] and Estimation of Distribution Algorithms [56]. They all share a common conceptual base of simulating the evolution of individual structures and they differ in the way the problem is represented, processes of selection and the usage/implementation of reproduction operators. The processes depend on the perceived performance of the individual structures as defined by the problem.

Compared to other global optimization techniques, evolutionary algorithms are easy to implement and very often they provide adequate solutions. A population of candidate solutions (for the optimization task to be solved) is initialized. New solutions are created by applying reproduction operators (mutation and/or crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine, which solutions are to be maintained into the next generation. The procedure is then iterated.

A. Abraham and C. Grosan: *Engineering Evolutionary Intelligent Systems: Methodologies, Architectures and Reviews*, Studies in Computational Intelligence (SCI) **82**, 1–22 (2008)
www.springerlink.com

The rest of the chapter is organized as follows. In Section 2, the various architectures for engineering evolutionary intelligent systems are presented. In Section 3, we present evolutionary artificial neural networks and its recent applications followed by evolutionary fuzzy systems and applications in Section 4. Evolutionary clustering is presented in Section 5 followed by recent applications of evolutionary design of complex paradigms in Section 6. Multiobjective evolutionary design intelligent systems are presented in Section 7 and some conclusions are provided towards the end.

2 Architectures of Evolutionary Intelligent Systems

Hybridization of evolutionary algorithms with other intelligent paradigms is a promising research field and various architectures for Evolutionary Intelligent Systems (EIS) could be formulated as depicted in Figures 1–6. By problem, we refer to any data mining/optimization/function approximation type problem and Intelligent Paradigm (IP) refers to any computational intelligence techniques like neural network, machine learning schemes, fuzzy inference systems, clustering algorithms etc.

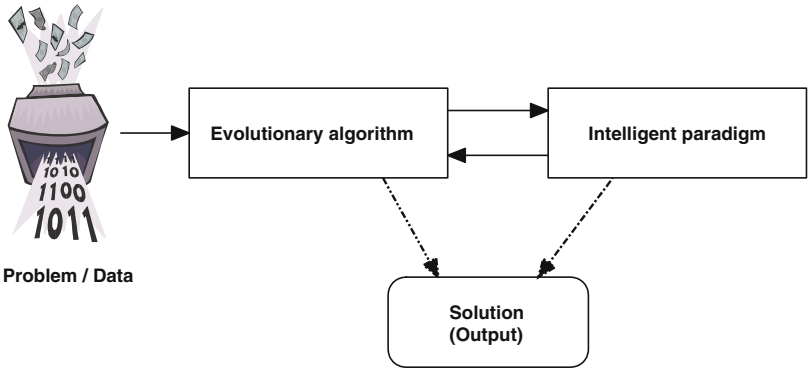

Fig. 1. EIS architecture 1

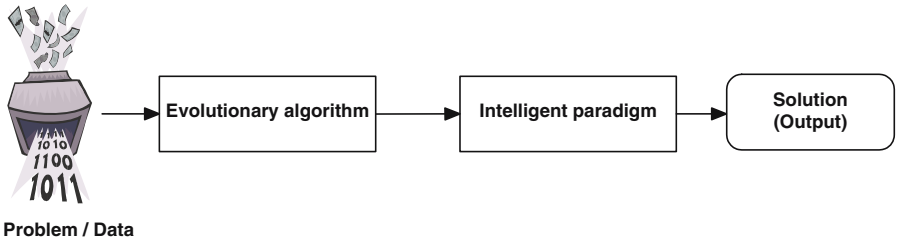

Fig. 2. EIS architecture 2

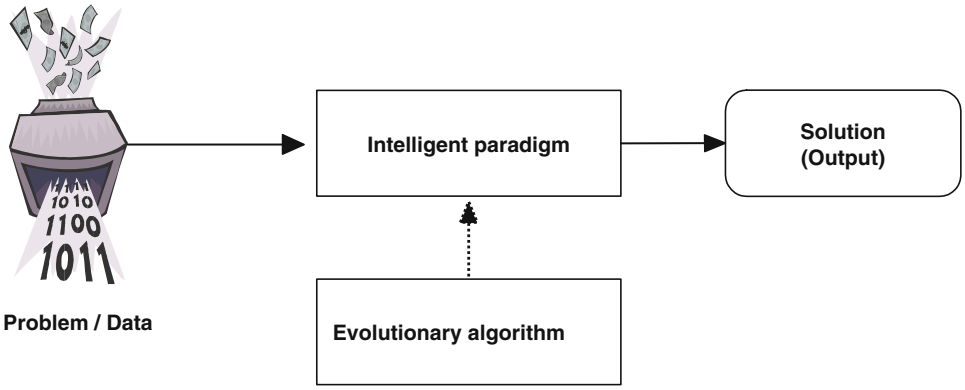

Fig. 3. EIS architecture 3

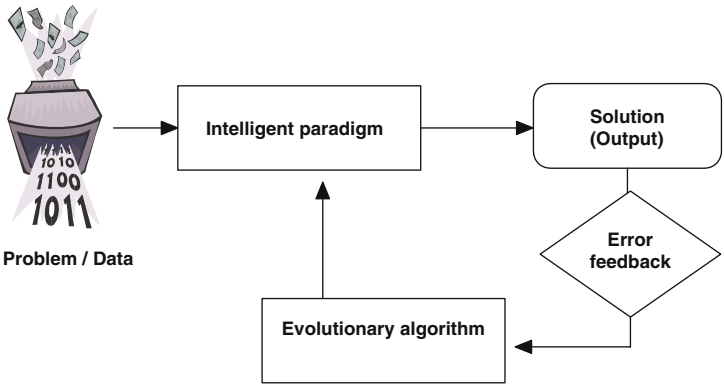

Fig. 4. EIS architecture 4

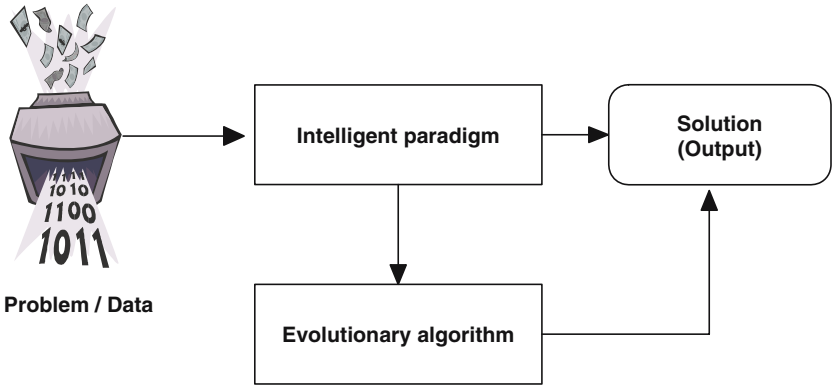

Fig. 5. EIS architecture 5

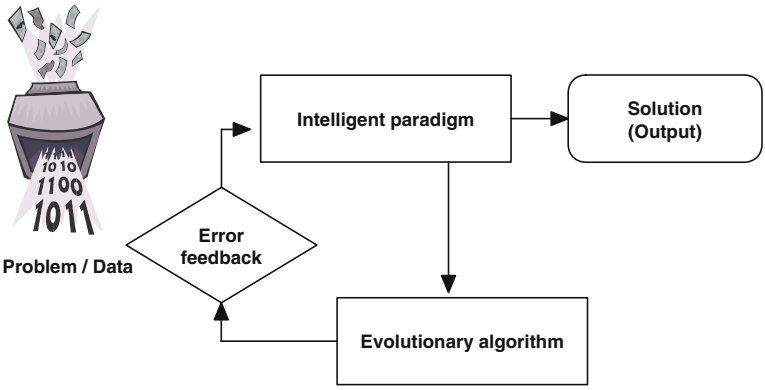

Fig. 6. EIS architecture 6

Figure 1 illustrates a transformational architecture where an evolutionary algorithm is used to optimize an intelligent paradigm and at the same time the intelligent paradigm is used to fine tune the parameters and performance of the evolutionary algorithm. An example is an evolutionary - fuzzy system where an evolutionary algorithm is used to fine tune the parameters of a fuzzy inference system (for a function approximation problem) and the fuzzy system is used to control the parameters of the evolutionary algorithm [32].

A concurrent hybrid architecture is depicted in Figure 2, where an EA is used as a pre-processor and the intelligent paradigm is used to fine tune the solutions formulated by the EA. The final solutions to the problem is provided by IP. Both EA and IP are continuously required for the satisfactory performance of the system. EA may be used as a post processor as illustrated in [1], [4] and [28].

Architecture 3 (Figure 3) depicts a cooperative hybrid system where the evolutionary algorithm is used to fine tune the parameters of IP only during the initialization of the system. Once the system is initialized the EA is not required for the satisfactory functioning of the system.

Architecture 4 uses an error feed back from the output (performance measure) and based on the error measure (critic information) the EA is used to fine tune the performance of the IP. Final solutions are provided by the the IP as illustrated in Figure 4.

An ensemble model is depicted in Figure 5 where EA received inputs directly from the IP and independent solutions are provided by the EA and IP. A slightly different architecture is depicted in Figure 6, where an error feed back is generated by the EA and depending on this input IP performance is fine tuned and final solutions are provided by the IP.

In the following Section, some of the well established hybrid frameworks for optimizing the performance of evolutionary algorithm using intelligent paradigms are presented.

3 Evolutionary Artificial Neural Networks

Artificial neural networks are capable of performing a wide variety of tasks, yet in practice sometimes they deliver only marginal performance. Inappropriate topology selection and learning algorithm are frequently blamed. There is little reason to expect that one can find a uniformly best algorithm for selecting the weights in a feedforward artificial neural network. This is in accordance with the no free lunch theorem, which explains that for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class [75].

At present, neural network design relies heavily on human experts who have sufficient knowledge about the different aspects of the network and the problem domain. As the complexity of the problem domain increases, manual design becomes more difficult and unmanageable. Evolutionary design of artificial neural networks has been widely explored. Evolutionary algorithms are used to adapt the connection weights, network architecture and learning rules according to the problem environment. A distinct feature of evolutionary neural networks is their adaptability to a dynamic environment. In other words, such neural networks can adapt to an environment as well as changes in the environment. The two forms of adaptation: evolution and learning in evolutionary artificial neural networks make their adaptation to a dynamic environment much more effective and efficient than the conventional learning approach.

In Evolutionary Artificial Neural Network (EANN), evolution can be introduced at various levels. At the lowest level, evolution can be introduced into weight training, where ANN weights are evolved. At the next higher level, evolution can be introduced into neural network architecture adaptation, where the architecture (number of hidden layers, no of hidden neurons and node transfer functions) is evolved. At the highest level, evolution can be introduced into the learning mechanism [77]. A general framework of EANNs which includes the above three levels of evolution is given in Figure 7 [2].

From the point of view of engineering, the decision on the level of evolution depends on what kind of prior knowledge is available. If there is more prior knowledge about EANN's architectures that about their learning rules or

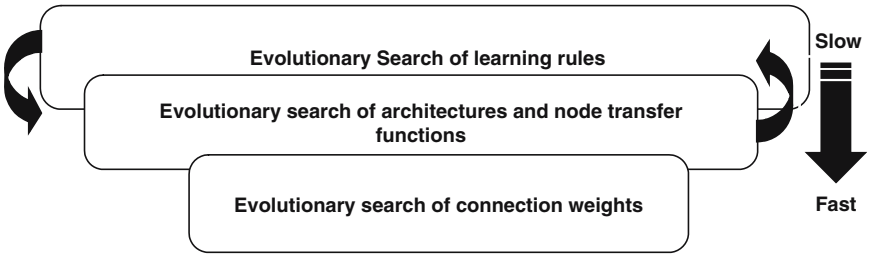

Fig. 7. A General framework for evolutionary artificial neural network

a particular class of architectures is pursued, it is better to implement the evolution of architectures at the highest level because such knowledge can be used to reduce the search space and the lower level evolution of learning rules can be more biased towards this kind of architectures. On the other hand, the evolution of learning rules should be at the highest level if there is more prior knowledge about them available or there is a special interest in certain type of learning rules.

3.1 Evolutionary Search of Connection Weights

The neural network training process is formulated as a global search of connection weights towards an optimal set defined by the evolutionary algorithm. Optimal connection weights can be formulated as a global search problem wherein the architecture of the neural network is pre-defined and fixed during the evolution. Connection weights may be represented as binary strings or real numbers. The whole network is encoded by concatenation of all the connection weights of the network in the chromosome. A heuristic concerning the order of the concatenation is to put connection weights to the same node together. Proper genetic operators are to be chosen depending upon the representation used. While gradient based techniques are very much dependant on the initial setting of weights, evolutionary search method can be considered generally much less sensitive to initial conditions. When compared to any gradient descent or second order optimization technique that can only find local optimum in a neighborhood of the initial solution, evolutionary algorithms always try to search for a global optimal solution. Evolutionary search for connection weights is depicted in Algorithm 3.1.

Algorithm 3.1 Evolutionary search of connection weights

1. Generate an initial population of N weight chromosomes. Evaluate the fitness of each EANN depending on the problem.

2. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.

3. Apply genetic operators to each child individual generated above and obtain the next generation.

4. Check whether the network has achieved the required error rate or the specified number of generations has been reached. Go to Step 2.

5. End

3.2 Evolutionary Search of Architectures

Evolutionary architecture adaptation can be achieved by constructive and destructive algorithms. Constructive algorithms, which add complexity to the network starting from a very simple architecture until the entire network is able to learn the task [23], [52]. Destructive algorithms start with large architectures and remove nodes and interconnections until the ANN is no longer able to perform its task [58], [66]. Then the last removal is undone. For an optimal network, the required node transfer function (Gaussian, sigmoidal, etc.) can be formulated as a global search problem, which is evolved simultaneously with the search for architectures [49].

To minimize the size of the genotype string and improve scalability, when priori knowledge of the architecture is known it will be efficient to use some indirect coding (high level) schemes. For example, if two neighboring layers are fully connected then the architecture can be coded by simply using the number of layers and nodes. The blueprint representation is a popular indirect coding scheme where it assumes architecture consists of various segments or areas. Each segment or area will define a set of neurons, their spatial arrangement and their efferent connectivity. Several high level coding schemes like graph generation system [44], Symbiotic Adaptive Neuro-Evolution (SANE) [54], marker based genetic coding [24], L-systems [10], cellular encoding [26], fractal representation [53], cellular automata [29] etc. are some of the rugged techniques.

Global search of transfer function and the connectivity of the ANN using evolutionary algorithms is formulated in Algorithm 3.2. The evolution of architectures has to be implemented such that the evolution of weight chromosomes are evolved at a faster rate i.e. for every architecture chromosome, there will be several weight chromosomes evolving at a faster time scale.

Algorithm 3.2 Evolutionary search of architectures

---

1. Generate an initial population of N architecture chromosomes. Evaluate the fitness of each EANN depending on the problem.

2. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.

3. Apply genetic operators to each child individual generated above and obtain the next generation.

4. Check whether the network has achieved the required error rate or the specified number of generations has been reached. Go to Step 2.

5. End

3.3 Evolutionary Search of Learning Rules

For the neural network to be fully optimal the learning rules are to be adapted dynamically according to its architecture and the given problem. Deciding the learning rate and momentum can be considered as the first attempt of learning rules [48]. The basic learning rule can be generalized by the function

$$\Delta w(t) = \sum_{k=1}^{n} \sum_{i_1, i_2, \ldots, i_k = 1}^{n} \left(\theta_{i_1, i_2, \ldots, i_k} \prod_{j=1}^{k} x_{ij}(t-1) \right) \tag{1}$$

where t is the time, Δw is the weight change, $x_1, x_2, \ldots . x_n$ are local variables and the θ's are the real values coefficients which is to be determined by the global search algorithm. In the above equation, different values of θ's determine different learning rules. The above equation is arrived based on the assumption that the same rule is applicable at every node of the network and the weight updating is only dependent on the input/output activations and the connection weights on a particular node.

The evolution of learning rules has to be implemented such that the evolution of architecture chromosomes are evolved at a faster rate i.e. for every learning rule chromosome, there will be several architecture chromosomes evolving at a faster time scale. Genotypes (θ's) can be encoded as real-valued coefficients and the global search for learning rules using the evolutionary algorithm is formulated in Algorithm 3.3.

In the literature, several research works could be traced about how to formulate different optimal learning rules [8], [21]. The adaptive adjustment of back-propagation algorithm parameters, such as the learning rate and momentum, through evolution could be considered as the first attempt of the evolution of learning rules [30]. Sexton et al. [65] used simulated annealing algorithm for optimization of learning. For optimization of the neural network

Algorithm 3.3 Evolutionary search of learning algorithms or rules

1. Generate an initial population of N learning rules. Evaluate the fitness of each EANN depending on the problem.

2. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.

3. Apply genetic operators to each child individual generated above and obtain the next generation.

4. Check whether the network has achieved the required error rate or the specified number of generations has been reached. Go to Step 2.

5. End

learning, in many cases a pre-defined architecture was used and in a few cases architectures were evolved together. Abraham [2] proposed the meta-learning evolutionary evolutionary neural network with a tight interaction of the different evolutionary search mechanisms using the generic framework illustrated in Figure 7.

3.4 Recent Applications of Evolutionary Neural Networks in Practice

Cai et al. [11] used a hybrid of Particle Swarm Optimization (PSO) [41], [18] and EA to train Recurrent Neural Networks (RNNs) for the prediction of missing values in time series data. Experimental results illustrate that RNNs, trained by the hybrid algorithm, are able to predict the missing values in the time series with minimum error, in comparison with those trained with standard EA and PSO algorithms.

Castillo et al. [12] explored several methods that combine evolutionary algorithms and local search to optimize multilayer perceptrons. Authors explored a method that optimizes the architecture and initial weights of multilayer perceptrons, a search algorithm for training algorithm parameters, and finally, a co-evolutionary algorithm, that handles the architecture, the network's initial weights and the training algorithm parameters. Experimental results show that the co-evolutionary method obtains similar or better results than the other approaches, requiring far less training epochs and thus, reducing running time.

Hui [37] proposed a new method for predicting the reliability for repairable systems using evolutionary neural networks. Genetic algorithms are used to globally optimize the number of neurons in the hidden layer and learning parameters of the neural network architecture.

Marwala [51] proposed a Bayesian neural network trained using Markov Chain Monte Carlo (MCMC) and genetic programming in binary space within Metropolis framework. The proposed algorithm could learn using samples obtained from previous steps merged using concepts of natural evolution which include mutation, crossover and reproduction. The reproduction function is the Metropolis framework and binary mutation as well as simple crossover, are also used.

Kim and Cho [42] proposed an incremental evolution method for neural networks based on cellular automata and a method of combining several evolved modules by a rule-based approach. The incremental evolution method evolves the neural network by starting with simple environment and gradually making it more complex. The multi-modules integration method can make complex behaviors by combining several modules evolved or programmed to do simple behaviors.

Kim [43] explored a genetic algorithm approach to instance selection in artificial neural networks when the amount of data is very large. GA optimizes

simultaneously the connection weights and the optimal selection of relevant instances.

Capi and Doya [13] implemented an extended multi-population genetic algorithm (EMPGA), where subpopulations apply different evolutionary strategies for designing neural controllers in the real hardware of Cyber Rodent robot. The EMPGA subpopulations compete and cooperate among each other.

Bhattacharya et al. [7] used a meta-learning evolutionary artificial neural network in selecting the best Flexible Manufacturing Systems (FMS) from a group of candidate FMSs. EA is used to evolve the architecture and weights of the proposed neural network method. Further, a Back-Propagation (BP) algorithm is used as the local search algorithm. All the randomly generated architectures of the initial population are trained by BP algorithm for a fixed number of epochs. The learning rate and momentum of the BP algorithm have been adapted suiting the generated data of the MCDM problem.

4 Evolutionary Fuzzy Systems

A conventional fuzzy controller makes use of a model of the expert who is in a position to specify the most important properties of the process. Fuzzy controller consists of a fuzzification interface, which receives the current values of the input variables and eventually transforms to linguistic terms or fuzzy sets. The knowledge base contains information about the domains of the variables, and the fuzzy sets associated with the linguistic terms. Also a rule base in the form of linguistic control rules is stored in the knowledge base. The decision logic determines the information about the control variables with the help of the measured input values and knowledge base. The task of defuzzification interface is to create a crisp control value out of the information about the control variable of the decision logic by using a suitable transformation.

The usual approach in fuzzy control is to define a number of concurrent *if-then* fuzzy rules. Most fuzzy systems employ the inference method proposed by Mamdani [50] in which the rule consequence is defined by fuzzy sets and has the following structure:

$$if\ x\ is\ A_1\ and\ y\ is\ B_1\ then\ f = C \tag{2}$$

Takagi, Sugeno and Kang (TSK) [72] proposed an inference scheme in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of the crisp inputs rather than a fuzzy set and has the following structure:

$$if\ x\ is\ A_1\ and\ y\ is\ B_1,\ then\ f = p_1 x + q_1 y + r \tag{3}$$

In the literature, several research works related to evolutionary design of fuzzy system could be located [59], [62]. Majority of the works are concerned with the automatic design or optimization of fuzzy logic controllers either by

adapting the fuzzy membership functions or by learning the fuzzy *if-then* rules [55], [33]. Figure 8 shows the architecture of the adaptive fuzzy control system wherein the fuzzy membership functions and the rule bases are optimized using a hybrid global search procedure. An optimal design of an adaptive fuzzy control system could be achieved by the adaptive evolution of membership functions and the learning rules that progress on different time scales. Figure 9 illustrates the general interaction mechanism with the global search of fuzzy rules evolving at the highest level on the slowest time scale. For each fuzzy rule base, global search of membership functions proceeds at a faster time scale in an environment decided by the problem.

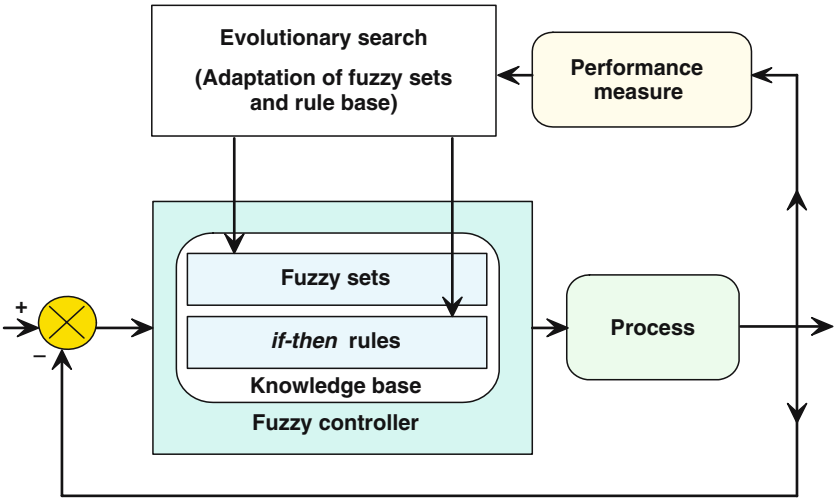

Fig. 8. Adaptive fuzzy control system architecture

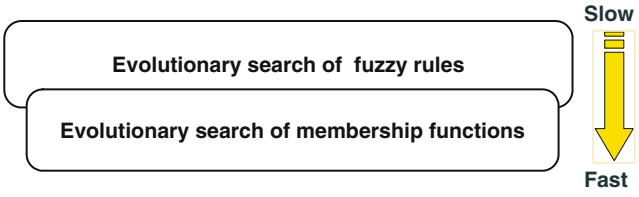

Fig. 9. Interaction of various search mechanisms in the design of optimal adaptive fuzzy control system

4.1 Evolutionary Search of Fuzzy Membership Functions

The tuning of the scaling parameters and fuzzy membership functions (piecewise linear and/or differentiable functions) is an important task in the design of fuzzy systems and is popularly known as genetic tuning. Evolutionary algorithms could be used to search the optimal shape, number of membership functions per linguistic variable and the parameters [31]. The genome encodes parameters of trapezoidal, triangle, logistic, Laplace, hyperbolic-tangent or Gaussian membership functions etc. Most of the existing methods assume the existence of a predefined collection of fuzzy membership functions giving meaning to the linguistic labels contained in the rules (database). Evolutionary algorithms are applied to obtain a suitable rule base, using chromosomes that code single rules or complete rule bases. If prior knowledge of the membership functions is available, a simplified chromosome representation could be formulated accordingly.

The first decision a designer has to make is how to represent a solution in a chromosome structure. First approach is to have the chromosome encode the complete rule base. Each chromosome differs only in the fuzzy rule membership functions as defined in the database. In the second approach, each chromosome encodes a different database definition based on the fuzzy domain partitions. The global search for membership functions using evolutionary algorithm is formulated in Algorithm 4.1.

4.2 Evolutionary Search of Fuzzy Rule Base

The number of rules grows rapidly with an increasing number of variables and fuzzy sets. Literature scan reveals that several coding methods were used

---

Algorithm 4.1 Evolution of learning of fuzzy membership functions and its parameters

---

1. Generate an initial population of N chromosomes using one of the approaches mentioned in Section 4.1. Evaluate the fitness of each fuzzy rule base depending on the problem.

2. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.

3. Apply genetic operators to each child individual generated above and obtain the next generation.

4. Check whether the fuzzy system has achieved the required error rate or the specified number of generations has been reached. Go to Step 2.

5. End

---

according to the nature of the problem. The rule base of the fuzzy system may be represented using relational matrix, decision table and set of rules.

In the *Pittsburg* approach, [71] each chromosome encodes a whole rule set. Crossover serves to provide a new combination of rules and mutation provides new rules. The disadvantage is the increased complexity of search space and additional computational burden especially for online learning. The size of the genotype depends on the number of input/output variables and fuzzy sets.

In the *Michigan* approach, [35] each genotype represents a single fuzzy rule and the entire population represents a solution. The fuzzy knowledge base is adapted as a result of antagonistic roles of competition and cooperation of fuzzy rules. A classifier rule triggers whenever its condition part matches the current input, in which case the proposed action is sent to the process to be controlled. The fuzzy behavior is created by an activation sequence of mutually collaborating fuzzy rules. In the Michigan approach, techniques for judging the performance of single rules are necessary.

The *Iterative Rule Learning* (IRL) approach [27] is similar to the Michigan approach where the chromosomes encode individual rules. In IRL, only the best individual is considered as the solution, discarding the remaining chromosomes in the population. The evolutionary algorithm generates new classifier rules based on the rule strengths acquired during the entire process.

Defuzzification operators and its parameters may be also formulated as an evolutionary search [46], [40], [5].

4.3 Recent Applications of Evolutionary Fuzzy Systems in Practice

Tsang et al. [73] proposed a fuzzy rule-based system for intrusion detection, which is evolved from an agent-based evolutionary framework and multi-objective optimization. The proposed system can also act as a genetic feature selection wrapper to search for an optimal feature subset for dimensionality reduction.

Edwards et al. [19] modeled the complex export pattern behavior of multinational corporation subsidiaries in Malaysia using a Takagi-Sugeno fuzzy inference system. The proposed fuzzy inference system is optimized by using neural network learning and evolutionary computation. Empirical results clearly show that the proposed approach could model the export behavior reasonably well compared to a direct neural network approach.

Chen et al. [15] proposed an automatic way of evolving hierarchical Takagi - Sugeno Fuzzy Systems (TS-FS). The hierarchical structure is evolved using Probabilistic Incremental Program Evolution (PIPE) with specific instructions. The fine tuning of the *if - then* rules parameters encoded in the structure is accomplished using Evolutionary Programming (EP). The proposed method interleaves both PIPE and EP optimizations. Starting with random structures and rules parameters, it first tries to improve the hierarchical structure and then as soon as an improved structure is found, it further fine tunes the rules parameters. It then goes back to improve the structure

and the rules' parameters. This loop continues until a satisfactory hierarchical TS-FS model is found or a time limit is reached.

Pawara and Ganguli [61] developed a Genetic Fuzzy System (GFS) for online structural health monitoring of composite helicopter rotor blades. Authors formulated a global and local GFSs. The global GFS is for matrix cracking and debonding/delamination detection along the whole blade and the local GFS is for matrix cracking and debonding/delamination detection in various parts of the blade.

Chua et al. [16] proposed a GA-based fuzzy controller design for tunnel ventilation systems. Fuzzy Logic Control (FLC) method has been utilized due to the complex and nonlinear behavior of the system and the FLC was optimized using the GA.

Franke et al. [22] presented a genetic - fuzzy system for automatically generating online scheduling strategies for a complex objective defined by a machine provider. The scheduling algorithm is based on a rule system, which classifies all possible scheduling states and assigns a corresponding scheduling strategy. Authors compared two different approaches. In the first approach, an iterative method is applied, that assigns a standard scheduling strategy to all situation classes. In the second approach, a symbiotic evolution varies the parameter of Gaussian membership functions to establish the different situation classes and also assigns the appropriate scheduling strategies.

5 Evolutionary Clustering

Clustering means the act of partitioning an unlabeled dataset into groups of similar objects. Each group, called a 'cluster', consists of objects that are similar between themselves and dissimilar to objects of other groups. A comprehensive review of the state-of-the-art clustering methods can be found in [76], [64].

Data clustering is broadly based on two approaches: hierarchical and partitional. In hierarchical clustering, the output is a tree showing a sequence of clustering with each cluster being a partition of the data set. Hierarchical algorithms can be agglomerative (bottom-up) or divisive (top-down). Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Partitional clustering algorithms, on the other hand, attempt to decompose the data set directly into a set of disjoint clusters by optimizing certain criteria. The criterion function may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically, the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters. The advantages of the hierarchical algorithms are the disadvantages of the partitional algorithms and vice versa.

Clustering can also be performed in two different modes: crisp and fuzzy. In crisp clustering, the clusters are disjoint and non-overlapping in nature. Any pattern may belong to one and only one class in this case. In case of fuzzy clustering, a pattern may belong to all the classes with a certain fuzzy membership grade.

One of the widely used clustering methods is the fuzzy c-means (FCM) algorithm developed by Bezdek [9]. FCM partitions a collection of n vectors $x_i,\ i = 1, 2 \ldots, n$ into c fuzzy groups and finds a cluster center in each group such that a cost function of dissimilarity measure is minimized. To accommodate the introduction of fuzzy partitioning, the membership matrix U is allowed to have elements with values between 0 and 1. The FCM objective function takes the form:

$$J(U, c_1, \ldots c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \tag{4}$$

where u_{ij}, is a numerical value between [0,1]; c_i is the cluster center of fuzzy group i; $d_{ij} = \|c_i - x_j\|$ is the Euclidian distance between i^{th}cluster center and j^{th} data point; and m *is* called the exponential weight which influences the degree of fuzziness of the membership (partition) matrix. Usually a number of cluster centers are randomly initialized and the FCM algorithm provides an iterative approach to approximate the minimum of the objective function starting from a given position and leads to any of its local minima [3]. No guarantee ensures that FCM converges to an optimum solution (can be trapped by local extrema in the process of optimizing the clustering criterion). The performance is very sensitive to initialization of the cluster centers.

Research efforts have made it possible to view data clustering as an optimization problem. This view offers us a chance to apply EA for evolving the optimal number of clusters and their cluster centers. The algorithm is initialized by constraining the initial values to be within the space defined by the vectors to be clustered. An important advantage of the EA is its ability to cope with local optima by maintaining, recombining and comparing several candidate solutions simultaneously.

Abraham [3] proposed the concurrent architecture of a fuzzy clustering algorithm (to discover data clusters) and a fuzzy inference system for Web usage mining. A hybrid evolutionary FCM approach is proposed in this paper to optimally segregate similar user interests. The clustered data is then used to analyze the trends using a Takagi-Sugeno fuzzy inference system learned using a combination of evolutionary algorithm and neural network learning.

6 Recent Applications of Evolutionary Design of Complex Paradigms

Park et al. [60] used EA to optimize Hybrid Self-Organizing Fuzzy Polynomial Neural Networks (HSOFPNN)m, which are based on genetically optimized multi-layer perceptrons. The architecture of the resulting HSOFPNN

combines fuzzy polynomial neurons (FPNs) [57] that are located at the first layer of the network with polynomial neurons (PNs) forming the remaining layers of the network. The GA-based design procedure being applied at each layer of HSOFPNN leads to the selection of preferred nodes of the network (FPNs or PNs) whose local characteristics (such as the number of input variables, the order of the polynomial, a collection of the specific subset of input variables, the number of membership functions for each input variable, and the type of membership function) can be easily adjusted.

Juang and Chung [39] proposed a recurrent TakagiSugenoKang (TSK) fuzzy network design using the hybridization of a multi-group genetic algorithm and particle swarm optimization (R-MGAPSO). Both the number of fuzzy rules and the parameters in a TRFN are designed simultaneously by R-MGAPSO. In R-MGAPSO, the techniques of variable-length individuals and the local version of particle swarm optimization are incorporated into a genetic algorithm, where individuals with the same length constitute the same group, and there are multigroups in a population.

Aouiti et al. [6] proposed an evolutionary method for the design of beta basis function neural networks (BBFNN) and beta fuzzy systems (BFS). Authors used a hierarchical genetic learning model of the BBFNN and the BFS.

Chen at al. [14] introduced a new time-series forecasting model based on the flexible neural tree (FNT). The FNT model is generated initially as a flexible multi-layer feed-forward neural network and evolved using an evolutionary procedure. FNT model could also select the appropriate input variables or time-lags for constructing a time-series model.

7 Multiobjective Evolutionary Design of Intelligent Paradigms

Even though some real world problems can be reduced to a matter of single objective very often it is hard to define all the aspects in terms of a single objective. In single objective optimization, the search space is often well defined. As soon as there are several possibly contradicting objectives to be optimized simultaneously, there is no longer a single optimal solution but rather a whole set of possible solutions of equivalent quality. When we try to optimize several objectives at the same time the search space also becomes partially ordered. To obtain the optimal solution, there will be a set of optimal trade-offs between the conflicting objectives. A multiobjective optimization problem is defined by a function f which maps a set of constraint variables to a set of objective values.

Delgado and Pegalajar [17], developed a multi-objective evolutionary algorithm, which is able to determine the optimal size of recurrent neural networks for any particular application. Authors analyzed in the case of grammatical inference: in particular, how to establish the optimal size of a recurrent neural network in order to learn positive and negative examples in a certain language,

and how to determine the corresponding automaton using a self-organizing map once the training has been completed.

Serra and Bottura [70] proposed a gain scheduling adaptive control scheme based on fuzzy systems, neural networks and multiobjective genetic algorithms for nonlinear plants. A FLC is developed, which is a discrete time version of a conventional one. Its data base as well as the controller gains are optimally designed by using a genetic algorithm for simultaneously satisfying the overshoot and settling time minimizations and output response smoothing.

Kelesoglu [47] developed a method for solving fuzzy multiobjective optimization of space truss using GA. The displacement, tensile stress, fuzzy sets, membership functions and minimum size constraints are considered in formulation of the design problem.

Lin [48] proposed a multiobjective and multistage fuzzy competence set model using a hybrid genetic algorithm. Author illustrated that the proposed method can provide a sound fuzzy competence set model by considering the multiobjective and the multistage situations simultaneously.

Ishibuchi and Nojimaa [38] examined the interpretability-accuracy tradeoff in fuzzy rule-based classifiers using a multiobjective fuzzy genetics-based machine learning (GBML) algorithm which is a hybrid version of Michigan and Pittsburgh approaches. Each fuzzy rule is represented by its antecedent fuzzy sets as an integer string of fixed length. Each fuzzy rule-based classifier, which is a set of fuzzy rules, is represented as a concatenated integer string of variable length. The GBML algorithm simultaneously maximizes the accuracy of rule sets and minimizes their complexity. The accuracy is measured by the number of correctly classified training patterns while the complexity is measured by the number of fuzzy rules and/or the total number of antecedent conditions of fuzzy rules.

Garcia-Pedrajas et al. [25] developed a cooperative coevolutive model for the evolution of neural network topology and weights, called MOBNET. MOBNET evolves subcomponents that must be combined in order to form a network, instead of whole networks. The subcomponents in a cooperative coevolutive model must fulfill different criteria to be useful, these criteria usually conflict with each other. The problem of evaluating the fitness on an individual based on many criteria that must be optimized together is approached as a multi-criteria optimization problems.

Wang et al. [74] proposed a multiobjective hierarchical genetic algorithm (MOHGA) to extract interpretable rule-based knowledge from data. In order to remove the redundancy of the rule base proactively, authors applied an interpretability-driven simplification method. Fuzzy clustering is used to generate an initial rule-based model and then MOHGA and the recursive least square method are used to obtain the optimized fuzzy models.

Pettersson et al. [63] used an evolutionary multiobjective technique in the training process of a feed forward neural network, using noisy data from an industrial iron blast furnace. The number of nodes in the hidden layer, the architecture of the lower part of the network, as well as the weights used in

them were kept as variables, and a Pareto front was effectively constructed by minimizing the training error along with the network size.

8 Conclusions

This Chapter presented the various architectures for designing intelligent paradigms using evolutionary algorithms. The main focus was on designing evolutionary neural networks and evolutionary fuzzy systems. We also illustrated some of the recent generic evolutionary design architectures reported in the literature including fuzzy neural networks and multiobjective design strategies.

References

1. Abraham A, Grosan C, Han SY, Gelbukh A (2005) Evolutionary multiobjective optimization approach for evolving ensemble of intelligent paradigms for stock market modeling. In: Alexander Gelbukh et al. (eds.) 4th Mexican international conference on artificial intelligence, Mexico, Lecture notes in computer science, Springer, Berlin Heidelberg New York, pp 673–681
2. Abraham, A (2004) Meta-learning evolutionary artificial neural networks. Neurocomput J 56c:1–38
3. Abraham A (2003) i-Miner: A Web Usage Mining Framework Using Hierarchical Intelligent Systems, The IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'03, IEEE Press, ISBN 0780378113, pp 1129–1134
4. Abraham A, Ramos V (2003), Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming, 2003 IEEE Congress on Evolutionary Computation (CEC2003), Australia, IEEE Press, ISBN 0780378040, pp 1384–1391, 2003
5. Abraham A (2003), EvoNF: A Framework for Optimization of Fuzzy Inference Systems Using Neural Network Learning and Evolutionary Computation, The 17th IEEE International Symposium on Intelligent Control, ISIC'02, IEEE Press, ISBN 0780376218, pp 327–332
6. Aouiti C, Alimi AM, Karray F, Maalej A (2005) The design of beta basis function neural network and beta fuzzy systems by a hierarchical genetic algorithm. Fuzzy Sets Syst 154(2):251–274
7. Bhattacharya A, Abraham A, Vasant P, Grosan C (2007) Meta-learning evolutionary artificial neural network for selecting flexible manufacturing systems under disparate level-of-satisfaction of decision maker. Int J Innovative Comput Inf Control 3(1):131–140
8. Baxter J (1992) The evolution of learning algorithms for artificial neural networks, Complex systems, IOS, Amsterdam, pp 313–326
9. Bezdek, JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum, New York
10. Boers EJW, Borst MV, Sprinkhuizen-Kuyper IG (1995) Artificial neural nets and genetic algorithms. In: Pearson DW et al. (eds.) Proceedings of the international conference in Ales, France, Springer, Berlin Heidelberg New York, pp 333–336

11. Cai X, Zhang N, Venayagamoorthy GK, Wunsch II DC (2007) Time series prediction with recurrent neural networks trained by a hybrid PSOEA algorithm. Neurocomputing 70(13–15):2342–2353
12. Castillo PA, Merelo JJ, Arenas MG, Romero G (2007) Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters. Inf Sci 177(14):2884–2905
13. Capi G, Doya K (2005), Evolution of recurrent neural controllers using an extended parallel genetic algorithm. Rob Auton Syst 52(2–3):148–159
14. Chen Y, Yang B, Dong J, Abraham A (2005) Time-series forecasting using flexible neural tree model. Inf Sci 174(3–4):219–235
15. Chen Y, Yang B, Abraham A, Peng L (2007) Automatic design of hierarchical takagi-sugeno fuzzy systems using evolutionary algorithms. IEEE Trans Fuzzy Syst 15(3):385–397
16. Chu B, Kim D, Hong D, Park J, Chung JT, Chung JH, Kim TH (2008) GA-based fuzzy controller design for tunnel ventilation systems, Journal of Automation in Construction, 17(2):130–136
17. Delgado M, Pegalajar MC (2005) A multiobjective genetic algorithm for obtaining the optimal size of a recurrent neural network for grammatical inference. Pattern Recognit 38(9):1444–1456
18. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of 6th Internationl Symposium on Micro Machine and Human Science, Nagoya, Japan, IEEE Service Center, Piscataaway, NJ, pp 39–43
19. Edwards R, Abraham A, Petrovic-Lazarevic S (2005) Computational intelligence to model the export behaviour of multinational corporation subsidiaries in Malaysia. Int J Am Soc Inf Sci Technol (JASIST) 56(11):1177–1186
20. Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, USA
21. Fontanari JF, Meir R (1991) Evolving a learning algorithm for the binary perceptron, Network, vol. 2, pp 353–359
22. Franke C, Hoffmann F, Lepping J, Schwiegelshohn U (2008) Development of scheduling strategies with Genetic Fuzzy systems, Applied Soft Computing Journal, 8(1):706–721
23. Frean M (1990), The upstart algorithm: a method for constructing and training feed forward neural networks. Neural Comput 2:198–209
24. Fullmer B, Miikkulainen R (1992) Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In: Varela FJ, Bourgine P (eds.) Proceedings of the first European conference on artificial life, France, pp 255–262
25. Garca-Pedrajas N, Hervs-Martnez C, Muoz-Prez J (2002) Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). Neural Netw 15(10):1259–1278
26. Grau F (1992)Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In: Whitely D, Schaffer JD (eds.) Proceedings of the international workshop on combinations of genetic algorithms and neural Networks, IEEE Computer Society Press, CA, pp 55–74
27. Gonzalez A, Herrera F (1997) Multi-stage genetic fuzzy systems based on the iterative rule learning approach. Mathware Soft Comput 4(3)
28. Grosan C, Abraham A, Nicoara M (2005) Search optimization using hybrid particle sub-swarms and evolutionary algorithms. Int J Simul Syst, Sci Technol UK 6(10–11):60–79

29. Gutierrez G, Isasi P, Molina JM, Sanchis A, Galvan IM (2001) Evolutionary cellular configurations for designing feedforward neural network architectures, connectionist models of neurons. In: Jose Mira et al. (eds.) Learning processes, and artificial intelligence, Springer, Berlin Heidelberg New York, LNCS 2084, pp 514–521
30. Harp SA, Samad T, Guha A (1989) Towards the genetic synthesis of neural networks. In: Schaffer JD (ed.) Proceedings of the third international conference on genetic algorithms and their applications, Morgan Kaufmann, CA, pp 360–369
31. Herrera F, Lozano M, Verdegay JL (1995) Tuning fuzzy logic controllers by genetic algorithms. Int J Approximate Reasoning 12:299–315
32. Herrera F, Lozano M, Verdegay JL (1995) Tackling fuzzy genetic algorithms. In: Winter G, Periaux J, Galan M, Cuesta P (eds.) Genetic algorithms in engineering and computer science, Wiley, USA, pp 167–189
33. Hoffmann F (1999) The Role of Fuzzy Logic in Evolutionary Robotics. In: Saffiotti A, Driankov D (ed.) Fuzzy logic techniques for autonomous vehicle navigation, Springer, Berlin Heidelberg New York
34. Holland JH (1975) Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI
35. Holland JH, Reitman JS (1978), Cognitive systems based on adaptive algorithms. In: Waterman DA, Hayes-Roth F (eds.) Pattern-directed inference systems. Academic, San Diego, CA
36. Holland, JH (1980) Adaptive algorithms for discovering and using general patterns in growing knowledge bases. Int J Policy Anal Inf Sys 4(3):245–268
37. Hui LY (2007) Evolutionary neural network modeling for forecasting the field failure data of repairable systems. Expert Syst Appl 33(4):1090–1096
38. Ishibuchi H, Nojima Y (2007) Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. Int J Approximate Reason 44(1):4–31
39. Juang CF, Chung IF (2007) Recurrent fuzzy network design using hybrid evolutionary learning algorithms, Neurocomputing 70(16–18):3001–3010
40. Jin Y, von Seelen W (1999) Evaluating flexible fuzzy controllers via evolution strategies. Fuzzy Sets Syst 108(3):243–252
41. Kennedy J, Eberhart RC (1995). Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, pp 1942–1948
42. Kim KJ, Cho SB (2006) Evolved neural networks based on cellular automata for sensory-motor controller. Neurocomputing 69(16–18):2193–2207
43. Kim KJ (2006) Artificial neural networks with evolutionary instance selection for financial forecasting. Expert Syst Appl 30(3):519–526
44. Kitano H (1990) Designing neural networks using genetic algorithms with graph generation system. Complex Syst 4(4):461–476
45. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, MIT, Cambridge, MA
46. Kosinski W (2007) Evolutionary algorithm determining defuzzyfication operators. Eng Appl Artif Intell 20(5):619–627
47. Kelesoglu O (2007) Fuzzy multiobjective optimization of truss-structures using genetic algorithm. Adv Eng Softw 38(10):717–721

48. Lin CM (2006) Multiobjective fuzzy competence set expansion problem by multistage decision-based hybrid genetic algorithms. Appl Math Comput 181(2):1402–1416
49. Liu Y, Yao X (1996) Evolutionary design of artificial neural networks with different node transfer functions. In: Proceedings of the Third IEEE International Conference on Evolutionary Computation, Nagoya, Japan, pp 670–675
50. Mamdani EH, Assilian S (1975) An experiment in linguistic synthesis with a fuzzy logic controller. Int J Man Mach Stud 7(1):1–13
51. Marwala T (2007) Bayesian training of neural networks using genetic programming. Pattern Recognit Lett 28(12):1452–1458
52. Mascioli F, Martinelli G (1995) A constructive algorithm for binary neural networks: the oil spot algorithm. IEEE Trans Neural Netw 6(3):794–797
53. Merril JWL, Port RF (1991) Fractally configured neural networks. Neural Netw 4(1):53–60
54. Moriarty DE, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evol Comput 5:373–399
55. Mohammadian M, Stonier RJ (1994) Generating fuzzy rules by genetic algorithms. In: Proceedings of 3rd IEEE International Workshop on Robot and Human Communication, Nagoya, pp 362–367
56. Muhlenbein H, Paab G (1996) From recombination of genes to the estimation of distributions I. Binary parameters. In: Lecture notes in computer science 1411: parallel problem solving from nature-PPSN IV, pp 178–187
57. Oh SK, Pedrycz W, Roh SB (2006), Genetically optimized fuzzy polynomial neural networks with fuzzy set-based polynomial neurons. Inf Sci 176(23):3490–3519
58. Omlin CW, Giles CL (1993) Pruning recurrent neural networks for improved generalization performance. Techincal report No 93-6, CS Department, Rensselaer Institute, Troy, NY
59. Cordon O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases, World Scientific, Singapore, ISBN 981-02-4016-3, p 462
60. Park HS, Pedrycz W, Oh SK (2007) Evolutionary design of hybrid self-organizing fuzzy polynomial neural networks with the aid of information granulation. Expert Syst Appl 33(4):830–846
61. Pawar PM, Ganguli R (2007) Genetic fuzzy system for online structural health monitoring of composite helicopter rotor blades. Mech Syst Signal Process 21(5):2212–2236
62. Pedrycz W (ed.) (1997), Fuzzy evolutionary computation, Kluwer Academic Publishers, Boston, ISBN 0-7923-9942-0, p 336
63. Pettersson F, Chakraborti N, Saxen H (2007) A genetic algorithms based multi-objective neural net applied to noisy blast furnace data. Appl Soft Comput 7(1):387–397
64. Rokach L, Maimon O (2005) Clustering methods, data mining and knowledge discovery handbook, Springer, Berlin Heidelberg New York, pp 321–352
65. Sexton R, Dorsey R, Johnson J (1999) Optimization of neural networks: a comparative analysis of the genetic algorithm and simulated annealing. Eur J Oper Res 114:589–601
66. Stepniewski SW, Keane AJ (1997) Pruning back-propagation neural networks using modern stochastic optimization techniques. Neural Comput Appl 5:76–98

67. Storn R, Price K (1997) Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. J Global Optim 11(4):341–359
68. Rechenberg I, (1973) Evolutions strategie: optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Fromman-Holzboog, Stuttgart
69. Schwefel HP (1977) Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie, Birkhaeuser, Basel
70. Serra GLO, Bottura CP (2006) Multiobjective evolution based fuzzy PI controller design for nonlinear systems. Eng Appl Artif Intell 19(2):157–167
71. Smith SF (1980) A learning system based on genetic adaptive algorithms. PhD thesis, University of Pittsburgh
72. Takagi T, Sugeno M (1983) Derivation of fuzzy logic control rules from human operators control actions. In: Proceedings of the IFAC symposium on fuzzy information representation and decision analysis, pp 55–60
73. Tsang CH, Kwong S, Wang A (2007) Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognit 40(9):2373–2391
74. Wang H, Kwong S, Jin Y, Wei W, Man K (2005) A multi-objective hierarchical genetic algorithm for interpretable rule-based knowledge extraction. Fuzzy Sets Syst 149(1):149–186
75. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
76. Xu R, Wunsch D, (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678
77. Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):423–1447