Participatory Search in Evolutionary Fuzzy Modeling

Yi Ling Liu and Fernando Gomide

Abstract Search is one of the most useful procedures employed in numerous situations such as optimization, machine learning, information processing and retrieval. This chapter introduces participatory search, a class of population-based search algorithms constructed upon the participatory learning paradigm. Participatory search relies on search mechanisms that progress forming pools of compatible individuals. The individual that is the most compatible with the best individual is always kept in the current population. Random immigrants are added to complete the population at each algorithm step. Different types of recombination are possible. The first is a convex combination, arithmetic-like recombination modulated by the compatibility between individuals. The second is a recombination mechanism based on selective transfer. Mutation is an instance of differential variation modulated by compatibility between selected and recombined individuals. Applications concerning development of fuzzy rule-based models from actual data illustrate the potential of the algorithms. The performance of the models produced by participatory search algorithms are compared with a state of the art genetic fuzzy system. Experimental results show that the participatory search algorithm with arithmetic-like recombination performs better than the remaining ones.

1 Introduction

The interest in evolutionary procedures to develop fuzzy systems from data has gained considerable attention in the last decade. Evolutionary fuzzy systems are fuzzy systems with added evolutionary components. An important instance of evolutionary fuzzy systems is genetic fuzzy systems (GFS). GFS combine fuzzy systems

Y.L. Liu (✉) · F. Gomide
School of Electrical and Computer Engineering, University of Campinas,
Sao Paulo, Brazil
e-mail: yiling155@gmail.com

F. Gomide
e-mail: gomide@dca.fee.unicamp.br

D.A. Pelta and C. Cruz Corona (eds.), *Soft Computing Based Optimization and Decision Models*, Studies in Fuzziness and Soft Computing 360,
DOI 10.1007/978-3-319-64286-4_11

with genetic algorithms [1] to solve complex classification, approximation, nonlinear modeling and control problems.

As it is well known, genetic algorithm (GA) is a population-based stochastic search procedure whose idea is to evolve a population of individuals using selection, recombination, and mutation operations working in sequence during several steps called generations [2]. A fitness function distinguishes the ability of an individual to remain in the next population. The better the value of the fitness function achieved by an individual, the higher is its chance to survive. This is the survival of the fittest saga. Individuals, candidate solutions of a problem, are points in the search space. Differently from GA, differential evolution [3] creates new candidate solutions combining the existing ones via mutation, recombination, and selection working in sequence during several generations. DE keeps whichever candidate solution that achieves the highest performance.

In [4] we read the following: *In actual survival of the fittest saga, there appears to be additional processes going on. In particular, the objective function in addition to be determined by some external requirement is often affected by the population itself.*

An approach that has been devised mimic the effect that a population itself has in its evolution is participatory learning [5]. The key idea of participatory learning is to account for compatibility between observations and current state of the learner. As it will be shown late, selection and variation operators such as recombination and mutation can designed to account for the compatibility between the individuals of a population. Compatibility and similarity have been shown to be effective in evolutionary computation [6–9].

This chapter addresses a new class of population-based search algorithms based on participatory learning. In common with other types of evolutionary algorithms, participatory search operates with a population of solutions, rather than with a single solution at a step, and employs procedures to combine these solutions to create new ones. Participatory search algorithms are novel instances of evolutionary algorithms because they do not need to assume that evolutionary approaches must necessarily be based on randomization [10, 11] though they are compatible with randomized implementations. Participatory search algorithms embody principles that are still not used by other evolutionary approaches, and that prove advantageous to solve a variety of complex optimization and design tasks.

The performance of the participatory algorithms is evaluated using actual data and compared with a state of the art genetic fuzzy system approach developed in [1]. Computational results show that the participatory search algorithm with arithmetical-like recombination performs better than the GFS approach.

After this introduction the chapter proceeds as follows. Section 2 briefly reviews genetic fuzzy systems. Section 3 reminds the concept of participatory learning. Section 4 introduces the participatory search operators: selection, selective transfer, arithmetic-like recombination and mutation operators. The search algorithms summarized in Sect. 5. Section 6 evaluates the performance of the participatory search algorithms against state of the art genetic fuzzy systems approaches. Section 7 concludes the chapter and list issues that deserve further development.

2 Genetic Fuzzy Systems

This section gives a brief overview of genetic fuzzy systems (GFS) and their applications. The focus is on genetic fuzzy rule-based systems (GFRBS), one of the most important types of GFS. The structure of GFRBS is summarized in Fig. 1.

GFRBS is a fuzzy rule-based system enhanced by genetic algorithms. A fuzzy rule-based system (FRBS) is composed by a knowledge base (KB) that encodes the knowledge of a target model. The KB contains two main components, a data base and a fuzzy rule base. The data base (DB) stores data that characterize the linguistic variables used by the fuzzy rules, the membership functions that define the semantics of the linguistic labels, and the parameters of the model. The fuzzy rule base (RB) is a collection of fuzzy if-then rules. Other three components complete fuzzy rule-based models. The first is a fuzzification module to serve as an input interface with the fuzzy reasoning process. The second is an inference engine that performs fuzzy reasoning. The third is a defuzzification output interface module to convert a fuzzy output into a representative pointwise output. An effective approach to construct the KB of an FRBS is to simultaneously develop the DB and the RB within the same process, but in two steps such as in embedded GFRBS learning. Embedded GFRBS is a scheme to learn the DB using simultaneously a simple method to derive a RB for each DB.

Embedded GFRBS does not necessarily provide simple, transparent, and competitive models in terms of the generalization capability. They may not scale well in terms of processing time and memory, two essential requirements especially in high-dimensional, large-scale, and complex problem solving. These issues are addressed in [1] where a way to reduce the search space in an embedded genetic DB learning framework is suggested. Lateral displacement of fuzzy partitions using a unique parameter for all membership functions of each linguistic variable is one of the mechanisms adopted to reduce search space complexity. The idea is to prescreen promising partitions to avoid overfitting and to maintain coverage and semantic soundness

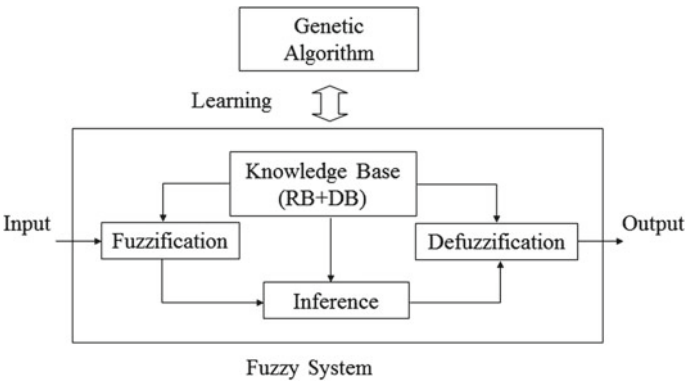


Fig. 1 Genetic fuzzy rule-based system

of the fuzzy partitions. The evolutionary algorithm also includes incest prevention, restarting, and rule-cropping in the RB generation process to improve convergence. Despite the use of mechanisms to manage dimensionality, the algorithm does not scale up on the number of data in datasets. A way to deal with scalability is to avoid large percentage of samples, and to estimate errors using a reduced subset. A post-processing step may further refine the algorithm.

Application examples of GFS are many. For example, [12] addresses a multi-objective optimization in which a fuzzy controller regulates the selection procedure and fitness function of genetic algorithms. Optimization is used to develop timetables of railway networks aiming at reducing passenger waiting time when switching trains, while at the same time, minimizing the cost of new investments to improve the necessary infrastructure. The result of the genetic optimization is a cost-benefit curve that shows the effect of investments on the accumulated passenger waiting time and trade-offs between both criteria. In [13] the aim is to optimize trip time and energy consumption of a high-speed railway with fuzzy c-means clustering and genetic algorithm. The method is used to develop a control strategy for a high-speed train line. An economical train runs with a trip time margin of less than 7% and an energy saving of 5% is reported. A model to relate the total length of low voltage line installed in a rural town with the number of people in the town and the mean of the distances from the center of the town to three furthest clients is discussed in [14]. The authors compare the training and test set error achieved by different modeling techniques for low line value estimation.

3 Participatory Learning

Participatory learning appeared in [5] as a process of learning that depends on what has already been learned. A central issue in the idea of participatory learning is that data have the greatest impact in causing learning or knowledge revision when they are compatible with the current knowledge. Learning occurs in an environment in which the current knowledge participates in the process of learning about itself. Clearly, a fundamental factor of participatory learning is the compatibility degree between input data and current knowledge. The current knowledge, denoted by $v(t)$, in addition to provide a standard against which input data $z(t)$ is compared with, directly affects the learning process. This is the participatory nature of learning process. High compatibility between the current knowledge and current input data opens the system for learning. In PL, this enhancement is expressed by the compatibility degree. A facility is provided to measure the confidence in the current knowledge structure. If a long sequence of input data have low compatibility with current knowledge, it may be the case that what has been learned so far is mistaken, not the data. This is seen as a form of stimulation called arousal. Participatory learning includes an arousal mechanism to monitor the performance of the learning process by watching at the values of the compatibility degrees of the current knowledge with

inputs. Monitoring information is fed back in terms of an arousal index that subsequently affects the learning process.

The instance of participatory learning we explore in this chapter uses the compatibility degree between current knowledge and current input data to update knowledge employing the following procedure [5, 15]:

$$v(t+1) = v(t) + \alpha\rho_t(z(t) - v(t)) \tag{1}$$

where $v(t)$ and $z(t)$ are n-dimensional vectors that denote the current knowledge and current input data, respectively. Assume, without loss of generality, that $v(t), z(t) \in [0,1]^n$. The parameter $\alpha \in [0,1]$ is the basic learning rate and $\rho_t \in [0,1]$ is the compatibility degree between $v(t)$ and $z(t)$ at step t. The product of the basic learning rate by the compatibility degree produces the actual learning rate. If an input is far from the current knowledge, then the value of the corresponding compatibility degree is small and the input is filtered. The actual learning rate is lowered by the compatibility degree. This means that if input data are too conflicting with the current knowledge, then they are discounted [5]. Lower values of actual learning rates avoid fluctuations due to values of input data which do not agree with current knowledge. As it will be shown shortly, (1) induces one of the recombination operators of participatory search algorithms.

The mechanism to monitor compatibility degrees during learning is the arousal index. The arousal index enters in the basic PL update formula (1) as follows

$$v(t+1) = v(t) + \alpha\rho_t^{1-a_t}(z(t) - v(t)) \tag{2}$$

where $a_t \in [0,1]$ is the arousal index at t.

One way to compute the compatibility degree ρ at step t is

$$\rho_t = 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k(t) - v_k(t)|. \tag{3}$$

In (3) ρ_t is the complement of the average absolute difference between input information $z(t)$ and current knowledge $v(t)$. In a more general sense, ρ_t may be seen to be a measure of similarity between $z(t)$ and $v(t)$. If $\rho_t = 0$, then $v(t+1) = v(t)$ and the current input $z(t)$ is completely incompatible with the current knowledge $v(t)$. This condition means that the system is not open to any learning from the current information. On the other hand, if $\rho_t = 1$, then $v(t+1) = z(t)$. In this case input information is in complete agreement with the current knowledge and the system is fully open to learn.

Arousal can be seen as the complement of the confidence in the current knowledge. A simple procedure is to update the arousal index a at step t is

$$a_{t+1} = (1-\beta)a_t + \beta(1 - \rho_{t+1}) \tag{4}$$

where $\beta \in [0, 1]$ controls the rate of change of arousal. The higher a_t, the less confident is the learning system in current knowledge. If $\rho_{t+1} = 1$, then we have a highly compatible input and the arousal index decreases. On the other hand, if $\rho_{t+1} = 0$, then input information compatibility is low and the arousal index increases.

The notion of compatibility degree enters in participatory search algorithms during the formation of pools of individuals for selection, recombination, and mutation. The pools are assembled from two populations S^t and $S^{t'}$. The individuals of $S^{t'}$ are those of S^t which are the most compatibles, one to one. Selection uses compatibility to choose those individuals from the pool that are closer to current best individual. Recombination is done pairwise between individuals of the mating pool, modulated by their compatibility degrees and arousal indexes. Mutation adds a variation to the current best individual proportional to the difference between the selected and recombined individuals modulated by the corresponding compatibility degrees. The effect of compatibility is to encourage selection and recombination of similar mates from which good offspring are likely to be produced, as indicated in [9].

4 Participatory Search Operators

The main construct elements of search algorithms are the representation, search operators, fitness function, and initial solution. These elements are relevant for all types of population-based algorithms. The remaining element is the search strategy. Representation concerns encoding mechanisms that maps problems solutions to strings. Representations allow definitions of search operators and of the search space. The search strategy defines types of intensification and diversification mechanisms.

In what follows we assume that a populations is a finite set of strings.

4.1 Selection

Let S be a set of N strings of fixed length n, and $s, s' \in S$ be two individuals, s' distinct of s, such that

$$s' = argmax_{r \in S}(\rho(s, r)) \tag{5}$$

where

$$\rho(s, r) = 1 - \frac{1}{n} \sum_{k=1}^{n} |s_k - r_k|, \tag{6}$$

and $s = (s_1, s_2, .., s_N)$ and $r = (r_1, r_2, .., r_N)$. Expression (5) means that s' is the individual of S whose compatibility degree with s is the largest. This procedure is repeated in sequence for each individual s of S to assemble a corresponding pool

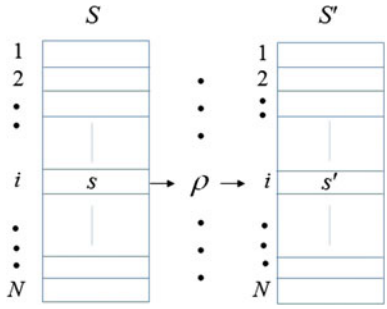


Fig. 2 A population and its pool of N individuals

$S^{'}$ with N individuals. Notice that construction of the pool is biased by the compatibility degrees between the individuals of S. Figure 2 illustrates how the populations S and S' are assembled.

In participatory search algorithms, selection is done by computing the compatibility degrees between $s \in S$ and the corresponding $s^{'} \in S^{'}$ with the current best individual $best = s^*$, and picking the one that is the most compatible to assemble a population L of selected individuals, that is, the ones that are the closest to the current best individual. Formally,

$$s^* = argmin_{s \in S} f(s), \tag{7}$$

where f is the objective function.

More specifically, selection computes the compatibility degrees $\rho^s(s, s^*)$ and $\rho^{s^{'}}(s^{'}, s^*)$ using

$$\rho^s = 1 - \frac{1}{n} \sum_{k=1}^{n} |s_k - s_k^*| \tag{8}$$

and

$$\rho^{s^{'}} = 1 - \frac{1}{n} \sum_{k=1}^{n} |s_k^{'} - s_k^*|, \tag{9}$$

and the individual whose compatibility degree is the largest, denoted by $p_{selected}$, is selected. That is, participatory selection proceeds according to the following rule

$$\text{if} \quad \rho^s \geq \rho^{s^{'}} \quad \text{then} \quad p_{selected} = s \quad \text{else} \quad p_{selected} = s^{'}. \tag{10}$$

Fig. 3 illustrates the process of selection.

Selection depends on the objective function $f(s)$, which identifies current best s^*, and on $\rho^s(s, s^*)$ and $\rho^{s^{'}}(s^{'}, s^*)$ which measure the compatibility between s^* and the corresponding pair of individuals s and $s^{'}$ of the current pool. Jointly, f, ρ^s and $\rho^{s^{'}}$ decide if an individual will be selected or not.

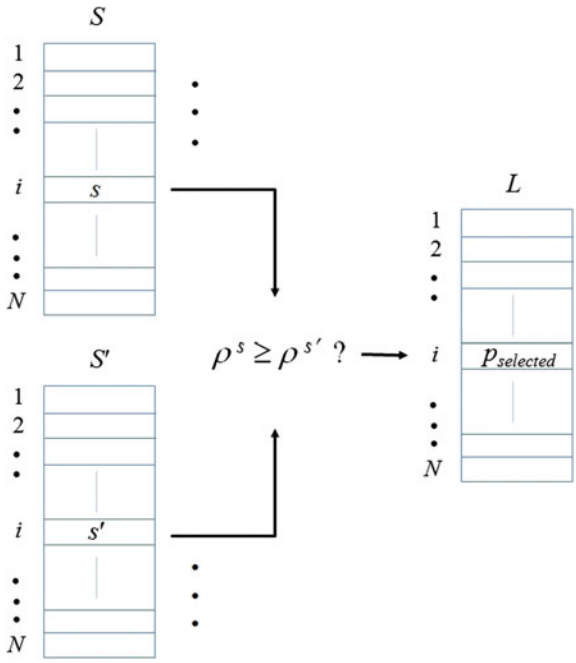


Fig. 3 Selection

4.2 Selective Transfer

During the last few years, we have witnessed a growing interest to use economic principles and models of learning in genetic algorithms. For instance, evolutionary processes have been used to model the adaptive behavior of a population of economic agents [16]. Here agents develop models of fitness to their environment in conjunction with the corresponding economic activities. Economists believe that behavior acquired through individual experience can be transmitted to future generations, and that learning changes the way to search the space in which evolution operates. This is an argument in favor of the interaction between the processes of evolution and learning. Since technical knowledge is distributed across the economic population, technological change can be viewed as a process of distributed learning. Here, the term learning is used in a broad sense, that is, there is no distinction between learning as propagation of knowledge through the population and the process of innovation, creation, and discovery. The distributed learning perspective helps to understand technological change and focus on the population suggests that an evolutionary perspective may be appropriate.

Birchenhall and Lin [16] claim that our knowledge and technology are modular, i.e., they can be decomposed into several components or modules. From the evolutionary computation point of view, they suggest that the crossover operator of genetic algorithms could be seen as a representative of modular imitation. To bring

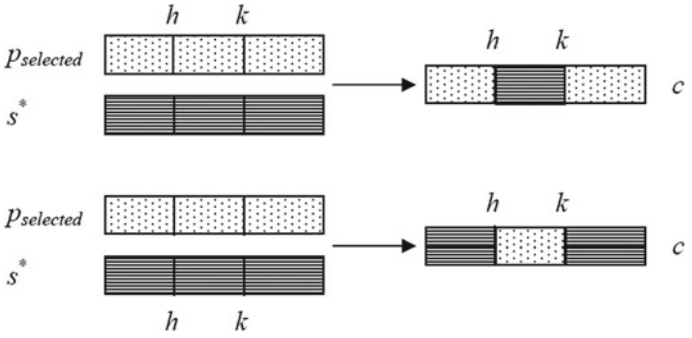


Fig. 4 Selective transfer

these ideas together, they advocate an algorithm that replaces selection and crossover operators by an operator based on selective transfer. Essentially, selective transfer is a filtered replacement of substrings from one string to another, without excluding the possibility that the entire sequence is copied [17]. Clearly, the selective transfer is similar to Holland crossover, but it is one-way transfer of strings, not on exchange of strings. The behavior selective transfer is likely to be very different from the combination of selection and crossover.

Assume that an individual $p_{selected}$ is selected using the objective function and compatibility. Two positions $h \leq k$ in the $p_{selected}$ string are chosen randomly, and a fair coin is tossed. If the coin turns head, then the substrings from $p_{selected}(h)$ to $p_{selected}(k)$ of $p_{selected}$ is replaced by the corresponding substrings from $s^*(h)$ to $s^*(k)$ of s^*. If the coin turns up tail, then the substrings from $p_{selected}(1)$ to $p_{selected}(h-1)$ and from $p_{selected}(k+1)$ to $p_{selected}(n)$ are replaced by the corresponding substrings of s^*. These steps are repeated for all individuals of L. Figure 4 illustrates the idea of selective transfer.

Despite similarity with crossover of the standard genetic algorithms, there are some differences. The most important one is that selective transfer uses one-way relocation of substrings, from the best individual to the one selected, and hence it is not a crossover. This is important because selective transfer is much more schemata destructive than the standard crossover [17].

4.3 Arithmetic Recombination

Arithmetic recombination emerges from the participatory learning update formula (2). To see this, notice that (2) can be rewritten as

$$\begin{aligned} v(t+1) &= v(t) + \alpha\rho_t^{(1-a_t)}(z(t) - v(t)) \\ &= (1 - \alpha\rho_t^{(1-a_t)})v(t) + \alpha\rho_t^{(1-a_t)}z(t). \end{aligned} \tag{11}$$

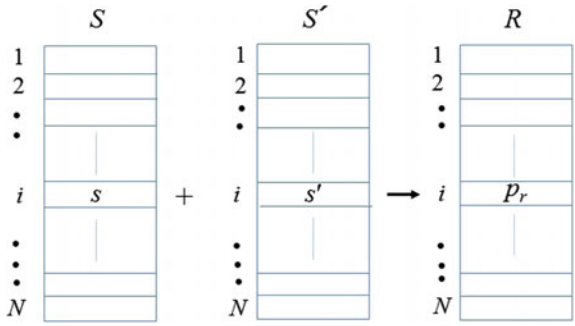


Fig. 5 Recombination

Let $\gamma = \alpha\rho_t^{(1-a_t)}$. Thus (11) becomes

$$v(t+1) = (1-\gamma)v(t) + \gamma z(t). \tag{12}$$

Expression (12) is of the following type

$$s_v(t+1) = (1-\delta)s_v(t) + \delta s_z(t) \tag{13}$$

where $\delta \in [0,1]$. Notice that (13) is a convex combination of $s_v(t)$ and $s_z(t)$ whose result is the offspring $s_v(t+1)$. Interestingly (12) is similar to (13) and hence (12) is an arithmetic-like recombination. While parameter δ of (13) is either a constant or variable, depending on the age of population, the value γ of (12) is variable and modulated by compatibility and arousal.

Participatory recombination proceeds as in (12) to produce offspring p_r from individuals s and s' of pools S and S', respectively, as follows

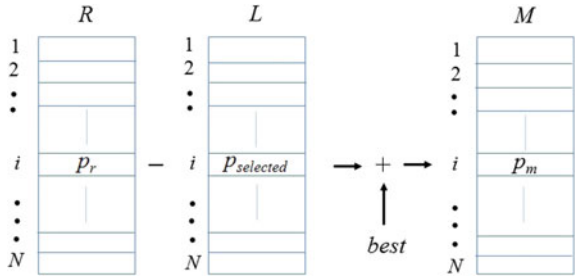
$$p_r = (1-\alpha\rho_r^{(1-a)})s + \alpha\rho_r^{(1-a)}s'. \tag{14}$$

Figure 5 illustrates the process of participatory recombination. Sums in the figure are done on an individual basis, and should be understood from the point of view of the operation (14).

4.4 Mutation

There are many ways to do mutation in search algorithms. For example, consider a population of N individuals represented by n-dimensional vectors denoted by $s_{r_i,t}$ at generation t. Differential evolution, for instance, produces new individuals by adding the weighted differences between distinct vectors to a third vector [3]. For each vector $s_{r_i,t}$, $i = 1, 2, ..., N$, a mutated vector is generated using

Fig. 6 Mutation



$$s_{i,t+1} = s_{r_1,t} + \phi \cdot (s_{r_2,t} - s_{r_3,t}) \tag{15}$$

where $r_1, r_2, r_3 \in \{1, 2, ..., N\}$ are random indexes, and $\phi > 0$ is a parameter which controls the amount of the differential variation $(s_{r_2,t} - s_{r_3,t})$.

Mutation in participatory search is similar to differential evolution mutation. It produces a mutated individual p_m as follows

$$p_m = best + \rho_m^{1-a}(p_{selected} - p_r). \tag{16}$$

Fig. 6 illustrates the process of mutation.

In participatory mutation, the amount of the variation of the best individual $best = s^*$ is controlled by compatibility between the selected and recombined individuals, and the arousal index.

5 Participatory Search Algorithms

Let S^t be the set of N with strings of length n at step t. The participatory search algorithms (PSA) start with a population S^t at $t = 0$ with N randomly chosen individuals and, for each individual of S^t, the most compatible individual amongst the remaining ones is chosen to assemble the population $S^{t'}$ with N individuals. S^t and $S^{t'}$ form the mating pool. Next, the best individual s^* in the current population S^t, denoted by *best*, is chosen. For instance, for minimization problems *best* is such that

$$best = argmin_{s \in S^t} f(s). \tag{17}$$

Selection chooses, by looking at each individual of S^t and the corresponding mate in $S^{t'}$, the one which is the closest to *best*. Recombination is done pairwise between the individuals of the mating pool, weighted by their values of compatibility and arousal. Mutation uses the selected and recombined individuals to produce variations whose amount is weighted by compatibility and arousal as well. If a offspring is better than the current best individual, then it replaces the current *best*. Otherwise, if a mutated individual is better than current best individual, then it replaces the

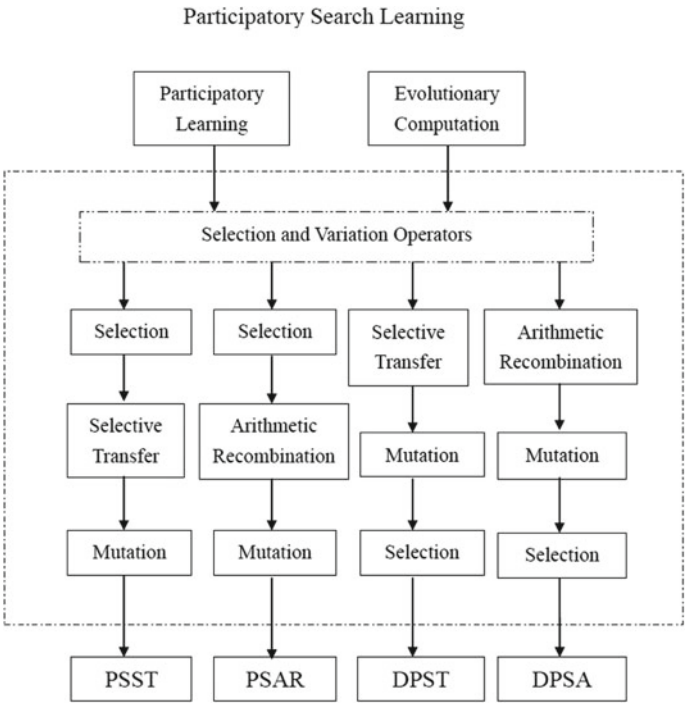


Fig. 7 Participatory search algorithms

current *best*. A new iteration starts with a new population S^{t+1} composed by the current best individual, with the remaining $(N-1)$ individuals chosen randomly. We should remark that participatory search algorithms are elitist: the best individual encountered is always kept in a population. The directive $last(S^t) \leftarrow best$ means that the best individual found up to generation t, denoted by *best*, is kept at the position that corresponds to the last individual of the population at step $t+1$.

There are four instances of PSA, respectively, participatory search with selective transfer (PSST), participatory search with arithmetic recombination (PSAR), differential participatory search with selective transfer (DPST), and differential participatory search with arithmetic recombination (DPSA). They are distinguished by the nature of the recombination, and the order in which the operations of selection, recombination, and mutation are processed in each generation. They also differ from similar evolutionary approaches developed in [6, 7, 18] in the way the mating pool is constructed to produce the new population. A class of participatory search algorithms that incorporates participatory learning is shown in Fig. 7.

PSST is similar to the algorithm discussed in [6] in the sense that both algorithms use participatory selective transfer and mutation. PSAR uses participatory arithmetic recombination and mutation, processed in a different order than PSST. DPST is similar to the algorithm of [7] because it also uses selective transfer and participatory

mutation. Likewise, DPSA is similar to the algorithm of [18] and uses participatory arithmetic recombination and mutation. DPSA proceeds similarly as DPST except that it uses arithmetic recombination instead of selective transfer. PSST, PSAR, DPST and DPSA differ from all previous approaches because selection is done individually for each of the N individuals of the current population. Participatory recombination and mutation are performed likewise. Recall that PSST, PSAR, DPST and DPSA are all elitist: the best individual is always kept in the current population. As an illustration, the procedure PSAR is detailed below. The remaining algorithms, except for their nature, have similar format. A in-depth description, characterization, and convergence analysis of the PSA can found in [19].

procedure PSAR
  f an objective function
  $s \in S^t$ and $s' \in S^{t'}$
  set *best* randomly
  set $a_0 \leftarrow 0$; $t \leftarrow 0$
  while $t \leq t_{max}$ **do**
    generate population S^t randomly
    $last(S^t) \leftarrow best$
    $S^{t'} \leftarrow s' = argmax_{r \in S^t}(\rho(s, r))$
    find *best* in S^t
Selection:
    compute $\rho^s(s, best)$ and $\rho^{s'}(s', best)$
    if $\rho^s \geq \rho^{s'}$ **then**
      $p_{selected} \leftarrow s$
    else
      $p_{selected} \leftarrow s'$
    end if
Recombination:
    choose $\alpha, \beta \in [0, 1]$ randomly
    compute $\rho_r = \rho(s, s')$
    compute $a_{t+1} = a_t + \beta((1 - \rho_r) - a_t)$
    $p_r = (1 - \alpha\rho_r^{1-a_t})s + \alpha\rho_r^{1-a_t}s'$
Mutation:
    compute $\rho_m = \rho(p_{selected}, p_r)$
    $p_m = best + \rho_m^{1-a_{t+1}}(p_{selected} - p_r)$
    if $f(p_r)$ better than $f(best)$ **then**
      $best \leftarrow p_r$
    end if
    if $f(p_m)$ better than $f(best)$ **then**
      $best \leftarrow p_m$
    end if
    $t \leftarrow t + 1$
  end while
  return *best*
end procedure

6 Participatory Search Algorithms in Fuzzy Modeling

This section concerns the use of participatory search algorithms in fuzzy rule-based system modeling. The aim is to illustrate potential applications of PSA and to evaluate and compare the performance of PSST, PSAR, DPST and DPSA algorithms using actual data and results reported in the literature.

The problem of interest here is to develop linguistic fuzzy models using actual data sets available in KEEL (http://www.keel.es/). The KEEL (Knowledge Extraction based on Evolutionary Learning) is a software tool to assess evolutionary algorithms for data mining problems including regression, classification, clustering, and pattern mining. KEEL provides a complete set of statistical procedures for multiple comparisons. The features of the data sets are summarized in Table 1. These data are the same used in [1], a state of the art representative GFS reported in the literature [20]. The representation and encoding schemes of PSAR are also the same of the one adopted in [1]. They are as follows:

1. Database encoding: $(C = C_1, C_2)$ a double-encoding scheme.
   First, equidistant strong fuzzy partitions are identified considering the granularity (labels) specified in C_1. Second, the membership functions of each variable are uniformly rearranged to a new position considering lateral displacement values specified in C_2.
   - Number of labels C_1: this is a vector of integers of size n representing the number of linguistic variables.

$$C_1 = (L^1, ..., L^n). \tag{18}$$

     Gene L^i is the number of labels of the ith linguistic variable, $L^i \in \{2, ..., 7\}$.
   - Lateral displacements C_2: this is a vector of real numbers of size n that encodes displacements α^i of the different variables, $\alpha^i \in [-0.1, 0.1]$. A detailed description of the linguistic 2-tuple representation is given in [21, 22].

Table 1 Summary of the datasets

Problem	Abbr.	Variables	Samples
Electrical maintenance	ELE	4	1056
Auto MPG6	MPG6	5	398
Analact	ANA	7	4052
Abalone	ABA	8	4177
Stock prices	STK	9	950
Forest fires	FOR	12	517
Treasury	TRE	15	1049
Baseball salaries	BAS	16	337

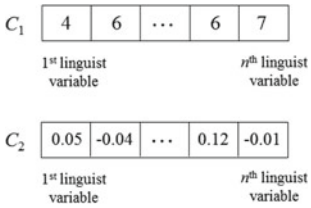


Fig. 8 A double-encoding scheme C_1 and C_2

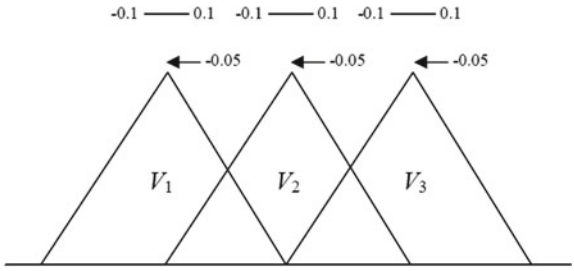


Fig. 9 Lateral displacement of the linguistic variable V values V_1, V_2, and V_3

$$C_2 = (\alpha^1, ..., \alpha^n). \tag{19}$$

An example of the encoding scheme is given in Fig. 8.
Figure 9 illustrates the lateral displacement of V for $\alpha = -0.05$.

2. Rule base: constructed using the Wang and Mendel algorithm (WM) [23, 24] as follows:

   a. granulate the input and output spaces;
   b. generate fuzzy rules using the given data;
   c. assign a certainty degree to each rule generated to resolve conflicts;
   d. create a fuzzy rule base combining the rules generated and rules provided by experts (if available);
   e. determine the input-output mapping using the combined fuzzy rule base and a defuzzification procedure.

   An example of a fuzzy rule-base developed for ELE is shown in Fig. 10.
   Example of rules of the rule base of Fig. 10 include:

rule 1: IF X1 is 1 and X2 is 1 and x3 is 1 and x4 is 1 THEN Y is 1
rule 2: IF X1 is 2 and X2 is 1 and x3 is 1 and x4 is 2 THEN Y is 3
rule 3: IF X1 is 3 and X2 is 3 and x3 is 2 and x4 is 3 THEN Y is 4

3. Objective function: the mean-squared error (*MSE*)

$$MSE = \frac{1}{2|D|} \sum_{l=1}^{|D|} (F(x^l) - y^l)^2 \tag{20}$$

Fig. 10 Rule base constructed using WM algorithm

 # Rule base : 28

X1	X2	X3	X4	Y
1	1	1	1	1
2	1	1	2	3
3	3	2	3	4
3	2	1	3	3
3	4	3	2	4
1	2	1	1	1
1	1	1	2	2
1	2	1	2	2
2	2	2	3	3
4	2	1	2	3
..	..	..	..	..

where $|D|$ is the size of the dataset, $F(x)$ is the output of the FRBS model, and y the actual value of the output. Fuzzy inference uses the max-min procedure with center of gravity deffuzification.

4. Initial population: each chromosome has the same number of linguistic labels, from two to seven labels for each input variable. For each label of the inputs, all possible combinations are assigned to the respective rules consequents. Moreover, for each combination, two copies are added with different values in the C_2 part. The first has values randomly chosen in $[-0.1, 0]$ and the second random values chosen in $[0, 0.1]$.
5. Recombination: $p_r \leftarrow floor(p_r)$ for C_1.
   If a gene g of p_r in C_1 is lower than 2, then $L_g = 2$, else if a gene g is higher than 7, then $L_g = 7$.
6. Mutation: $p_m \leftarrow floor(p_m)$ for C_1.
   If a gene g of p_m in C_1 is lower than 2, then $L_g = 2$, else if a gene g is higher than 7, then $L_g = 7$.

The electric maintenance model has four input variables and one output variable. The ELE dataset contains electrical maintenance data and has 1056 samples. This is an instance in which we expect learning methods to develop large number of rules. ELE modeling involves a large search space [1]. The MPG data concerns city-cycle fuel consumption in miles per gallon (mpg), to be predicted in terms of one multivalued discrete and five continuous attributes. The MPG6 dataset has 398 samples. The categorical data (ANA) is one of the data sets used in the book *Analyzing Categorical Data* by Jeffrey S. Simonoff. It contains information about the decisions taken by a supreme court. The ANA dataset concerns seven variables and 4052 samples. Abalone age data come from physical measurements. The abalone model has eight input variables and one output variable. The abalone dataset (ABA) contains 4177 samples. The STK data provided are daily stock prices from January 1988 through October 1991, for ten aerospace companies. The task is to approximate the price

Table 2 Methods considered by the computational experiments [1]

Method	Type of learning
WM(3)	Rule base produced by WM, 3 linguistic labels for each variable
WM(5)	Rule base produced by WM, 5 labels for each variable
WM(7)	Rule base produced by WM, 7 labels for each variable
FSMOGFS	Gr. Lateral partition parameters, and rule base produced by WM
FSMOGFS+TUN	FSMOGFS + Tuning of MF parameters and rule selection by SPEA2
FSMOGFS^e+TUN^e	FSMOGFS+TUN including fast error estimation

of the 10th company given the prices of the rest. The STK has nine input variables and 950 samples. The FOR dataset has 12 variables and 517 samples. The aim is to predict the burned area of forest fires, in the northeast region of Portugal. The TRE contains the economic data information of USA and has 15 variables input and 1049 samples. The goal is to predict 1-Month Rate. The BAS contains the salaries of the set of Major League Baseball players and has 16 variables input and 337 samples. The task is to approximate the salary of each player. The datasets are available at http://sci2s.ugr.es/keel/index.php. The methods considered in [1] are summarized in Table 2. The method of Wang and Mendel (WM) is also a reference because all PSA and the GFS use it as a rule generation procedure during evolution. The participatory search algorithms were run using the datasets to compare their results with the ones produced by PSAR and results reported in the literature [1]. The processing times of the different methods in [1] were obtained using an Intel Core 2 Quad Q9550 2.83-GHz, 8 GB RAM. The processing times of participatory search algorithms reported here were obtained using an Intel Core 2 Quad Q8400 2.66GHz, 4 GB RAM.

The input parameters used by participatory search algorithms in the experiments reported in this section are: population size of 60, and maximum number of function evaluations of 1000. Data sets were randomly split into five folds, each partition containing 20% of the dataset. Four of these partitions are used for training and the remaining one is used for testing. The algorithms are run six times for each data partition using six distinct seeds.

The results show that the average mean-squared error for the test data achieved by the fuzzy models developed by PSAR, Table 6, is lower than the average mean-squared error of test data achieved by the FSMOGFS^e+TUN^e, except for ANA data. Also, the average mean-squared error for the test data achieved by DPSA is lower than the FSMOGFS^e+TUN^e. For the test data of ANA, FSMOGFS^e+TUN^e achieves the lowest *MSE* value. Considering the test data PSAR, with WM using different number of labels for each linguistic variable, is more accurate than when the number of linguistic labels for each linguistic variable is kept fixed, WM(3),WM(5) and WM(7), respectively. Thus, PSAR performs better than FSMOGFS^e+TUN^e from the point of view of the test data of *MSE*. Also, the standard deviation (SD) of test data for PSAR and FSMOGFS^e+TUN^e is better than WM(3),WM(5) and WM(7).

Table 3 Average rank of the algorithms

Algorithm	Friedman rank	p-value	H_0
WM(3)	7.3125		
WM(5)	6.25		
WM(7)	6		
FSMOGFS^e+TUN^e	3.75	1.38E-7	Rejected
PSAR	2.125		
PSST	4		
DPSA	2.3125		
DPST	4.25		

Table 4 Holm's Post-Hoc for $\epsilon = 0.05$.

Control algorithm: PSAR					
i	Algorithm	z value	p-value	ϵ/i	H_0
7	WM(3)	4.2355	2.3E-5	0.0071	Rejected
6	WM(5)	3.368	0.0007	0.0083	Rejected
5	WM(7)	3.1639	0.0015	0.01	Rejected
4	DPST	1.735	0.08273	0.0125	Rejected
3	PSST	1.5309	0.1257	0.0166	Rejected
2	FSMOGFS^e+TUN^e	1.3268	0.184573	0.025	Accepted
1	DPSA	0.153	0.8783	0.05	Accepted

Further analysis is pursued as suggested in [25, 26] to verify if there exist statistical differences among the performance of the algorithms. Recall that the confidence level is $\epsilon = 0.05$. Table 3 shows how PSAR and GFS are ranked. PSAR achieves the highest rank with 1.375. Also, recall that the null hypothesis H_0 is that PSAR and GFS algorithms are equivalent, that is, H_0 means that the rank of all algorithms are equal. If the hypothesis is rejected, then we conclude that the algorithms perform differently.

Iman-Davenport's test suggests that there are significant differences among the algorithms in all datasets once the null hypothesis is rejected (p-value $= 1.38\,E{-}7$). Thus the Holm post-hoc test is conducted with PSAR as the control algorithm. Table 4 shows that the Holm post-hoc test rejects the hypothesis concerning WM(3), WM(5), WM(7), DPST and PSST, but do not reject FSMOGFS^e+TUN^e and DPSA. Therefore, PSAR outperforms WM(3), WM(5), WM(7), DPST and PSST because the rank pf PSAR is the highest and rejects the hypothesis in the Holm test. We notice that the difference of the performance of FSMOGFS^e+TUN^e and DPSA is not statistically relevant because the null hypothesis is accepted.

Table 5 highlights, that for each dataset, the average processing time of FSMOGFS^e+TUN^e and PSAR in minutes and seconds. We notice the different complexity of the solutions generated during the evolutionary process. The com-

Table 5 Average runtime of the algorithms (minutes:seconds M:S)

Dataset	FSMOGFS^e+TUN^e	PSAR
ELE	00:42	00:45
MPG6	1:00	00:53
ANA	5:17	5:05
ABA	3:54	4:25
STK	1:31	1:12
FOR	1:07	00:40
TRE	00:46	1:02
BAS	00:58	1:01

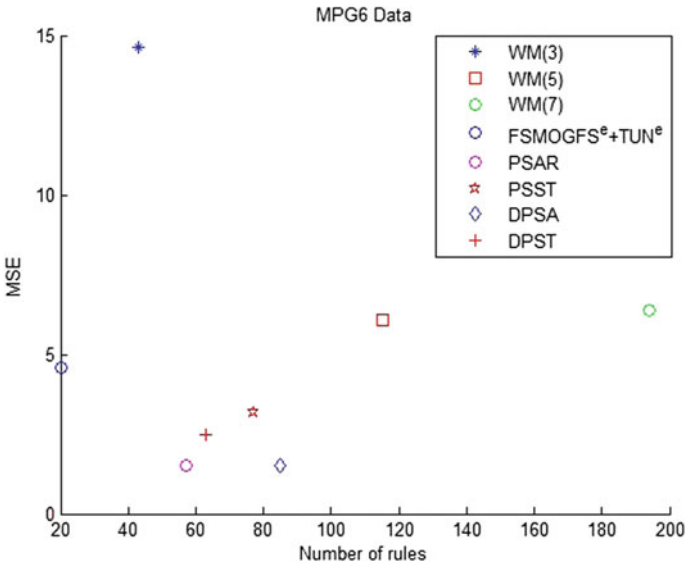


Fig. 11 MSE performance of the algorithms versus the number of rules: MPG6 data

putational cost of the fitness evaluation depends of the number of rules and conditions in rules antecedents. In the case of ANA, the PSAR needs less time than FSMOGFS^e+TUN^e because the number of rules is small. On the other hand, it is higher than 3 min in ANA and ABA because the large number of samples.

In sum, the performance of PSAR in developing fuzzy rule-based models with actual data illustrates its potential to solve complex problems. Overall, the results suggest that PSAR performs better than current state of the art genetic fuzzy system approaches from the point of view of the average mean square error with test data. Figure 11 summarizes the MSE performance of the algorithms versus the number of rules for MPG6 data set. More importantly, participatory search algorithms are

Table 6 Average *MSE* of PSA and GFS Algorithms

		WM(3)		WM(5)		WM(7)		FSMOGFS^e+TUN^e		PSAR		PSST		DPSA		DPST	
Dataset		Tra	Tst	Tra	Tst	Tra	Tst	Tra	Tst	Tra	Tst	Tra	Tst	Tra	Tst	Tra	Tst
ELE	Rule	27		65		103		8		28		27		23		23	
	Mean	192241	192647	56135	56359	53092	55495	9665	10548	9502	10480	11409	10560	9586	10434	11250	10544
	SD	9658	14436	1498	4685	1955	9452	823	1150	3951.94	3986.8	3874.74	3906.8	3914.4058	3753.2432	3020.37	3260.55
MPG6	Rule	43		115		194		20		57		77		85		63	
	Mean	13.552	14.649	4.136	6.096	2.642	6.382	2.86	4.562	1.6132	1.5205	3.424	3.1699	1.8901	1.5151	2.0641	2.467
	SD	1.239	3.204	0.317	2.416	0.11	2.126	0.218	0.714	1.3712	1.2983	1.3093	1.2639	1.1779	1.2563	1.1532	1.299
ANA	Rule	72		124		171		10		6		4		7		5	
	Mean	0.187	0.189	0.027	0.03	0.012	0.017	0.003	0.003	0.0256	0.0856	0.0292	0.0682	0.0274	0.0507	0.0292	0.0795
	SD	0.001	0.005	0	0.002	0	0.003	0	0.001	0.0833	0.0574	0.0875	0.0485	0.0913	0.0866	0.0836	0.0373
ABA	Rule	68		199		368		8		34		23		34		30	
	Mean	8.407	8.422	3.341	3.474	3.057	3.268	2.445	2.509	0.0016	0.0047	0.0018	0.005	0.0018	0.0047	0.0055	0.0048
	SD	0.443	0.545	0.13	0.247	0.084	0.185	0.114	0.184	0.0002	0.0002	0.0001	0.0002	0.0002	0.0002	0.0005	0.0002
STK	Rule	123		265		378		23		58		73		66		78	
	Mean	8.852	8.951	1.576	1.624	0.611	1.488	0.764	0.912	0.7342	0.9002	1.1022	0.9064	0.7784	0.8139	1.6701	1.8139
	SD	0.508	1.193	0.09	0.09	0.029	1.634	0.139	0.181	0.0479	0.1001	0.0687	0.1333	0.0912	0.0597	0.0738	0.0987
FOR	Rule	246		375		401		10		23		31		8		43	
	Mean	2030	3793	1435	34235	340	1.00E+05	1418	2628	2115.76	1619.13	2231.0333	2.35E+03	2652.8666	2.04E+03	1600.9333	1.74E+03
	SD	531	2340	505	4356	147	4708	539	2108	381.98	756.86	169.3108	450.3585	207.1127	461.7959	194.4706	524.3924
TRE	Rule	75		196		261		9		26		28		30		25	
	Mean	1.636	1.631	0.401	0.405	0.17	0.176	0.034	0.044	0.0529	0.0372	0.1168	0.1707	0.0655	0.0392	0.1492	0.128
	SD	0.121	0.181	0.014	0.055	0.009	0.017	0.003	0.015	0.0059	0.0045	0.0059	0.019	0.0067	0.008	0.0099	0.0205
BAS	Rule	181		253		264		17		50		52		47		65	
	Mean	1.9E+05	3.6E+05	7.8E+04	6.2E+05	3.1E+04	1.0E+06	1.4E+05	2.6E+05	1.5E+05	1.3E+05	1.0E+05	1.5E+05	1.1E+05	3.6E+05	1.0E+05	3.9E+05
	SD	1.0E+04	7.3E+04	4.7E+03	6.8E+04	6.0E+02	1.3E+05	1.9E+04	5.8E+04	7753.67	124393	5.9E+03	3.1E+04	7.6E+03	2.6E+05	7.2E+03	3.0E+05

simpler, have high computational performance, and require few parameters to run. In particular, PSAR is a highly competitive population-based search approach (Table 6).

7 Conclusion

Participatory search is a population-based instance of the participatory learning paradigm. Compatibility degrees and arousal indexes account for the effect of the population individuals during search. Recombination arises from an instance of participatory learning formula. The participatory search algorithms are elitist and employ compatibility and arousal information in selection, recombination and mutation. Applications concerned the use of participatory search algorithms to develop fuzzy linguistic models of actual data. The performance of the models produced by the participatory search algorithms were evaluated and compared with a state of the art genetic fuzzy system approach. The results suggest that the participatory search algorithm with arithmetical-like recombination performs best.

Acknowledgements The second author is grateful to CNPq, the Brazilian National Council for Scientific and Technological Development (CNPq), for grant 305906/2014-3.

References

1. Alcalá, R., Gacto, M., Herrera, F.: A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. IEEE Trans. Fuzzy Syst. **19**(4), 666–681 (2011)
2. Herrera, F., Verdegay, J.: Genetic Algorithms and Soft Computing. Physica-Verlag, Heidelberg, Germany (1996)
3. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
4. Yager, R.: Participatory Genetic Algorithms. BISC Group List, message posted on 29 Aug 2000
5. Yager, R.: A model of participatory learning. IEEE Trans. Syst. Man Cybern. **20**(5), 1229–1234 (1990)
6. Liu, Y.L., Gomide, F.: Participatory genetic learning in fuzzy system modeling. In: Proceedings of IEEE SSCI 2013, Singapore (2013)
7. Liu, Y.L., Gomide, F.: Evolutionary participatory learning in fuzzy system modeling. In: Proceedings of IEEE International Conference on Fuzzy System, p. 2013, Hyderabad, India (2013)
8. Liu, Y.L., Gomide, F.: Participatory search algorithms in fuzzy modeling.In: Proceedings of the World Conference in Soft Computing, Berkeley, USA (2016)
9. Ishibuchi, H., Narukawa, K., Tsukamoto, N., Nojima, Y.: An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization. Eur. J. Oper. Res. **188**, 57–75 (2008)
10. Fogel, D.: An Introduction to Evolutionary Computation, Chapter 1 of Evolutionary Computation: The Fossil Record. Wiley-IEEE Press, New York, USA (1998)
11. Glover, F., Marti, R.: Fundamentals of scatter search and path relinking. Control Cybern. **29**(3), 653–684 (2000)

12. Voget, S., Kolonko, M.: Multidimensional optimization with a fuzzy genetic algorithm. J. Heuristics **4**(3), 221–244 (1998)
13. Hwang, H.-S.: Control strategy for optimal compromise between trip time and energy consumption in a high-speed railway. IEEE Trans. Syst. **28**(6), 791–802 (1998)
14. Cordón, Ó.: Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases. In: Advances in Fuzzy Systems. World Scientific Publishing, Singapore (2001)
15. Brown, R.: Moothing, Forecasting and Prediction of Discrete Time Series. Prentice-Hall, New Jersey, USA (2004)
16. Birchenhall, C., Lin, J.-s.: Learning and adaptive artificial agents: Analysis of an evolutionary economic model. In: Computing in Economics and Finance, vol. 327. University of Manchester, United Kingdom (2000)
17. Hwang, H.-S.: Genetic algorithms in evolutionary modelling. J. Evolut. Econ. **7**(4), 375–393 (1997)
18. Liu, Y.L., Gomide, F.: Fuzzy systems modeling with participatory evolution. In: IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS). Edmonton, AB, Canada (2013)
19. Liu, Y.L.: Participatory search algorithms and applications. Ph.D. Thesis, School of Electrical and Computer Engineering, University of Campinas, Campinas, Sao Paulo, Brazil (2016)
20. Fazzolari, M., Alcala, R., Nojima, Y., Ishibuchi, H., Herrera, F.: A review of the application of multiobjective evolutionary fuzzy systems: current status and further directions. IEEE Trans. Fuzzy Syst. **21**, 45–65 (2013)
21. Herrera, F., Martínez, L.: A 2-tuple fuzzy linguistic representation model for computing with words. IEEE Trans. Fuzzy Syst. **8**(6), 746–752 (2000)
22. Alcalá, R., Alcalá-Fdez, J., Herrera, F., Otero, J.: Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. Int. J. Approx. Reason. **44**(1), 45–64 (2007)
23. Wang, L., Mendel, J.: Generation fuzzy rules by learning from examples. IEEE Trans. Syst. Man Cybern. **22**(6), 1414–1427 (1992)
24. Wang, L.: Adaptive Fuzzy Systems and Control: Design and Stability Analysis. Prentice-Hall, Upper Saddle River, USA (1994)
25. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut. Comput. **1**, 3–18 (2011)
26. Antonelli, M., Ducange, P., Marcelloni, F.: An efficient multi-objective evolutionary fuzzy system for regression problems. Int. J. Approx. Reason. **54**, 1434–1451 (2013)