Soft Comput (2016) 20:3303–3319
DOI 10.1007/s00500-015-1708-3

Sifter: an approach for robust fuzzy rule set discovery

Ali Mohammadi Shanghooshabad[1] · **Mohammad Saniee Abadeh**[1]

Published online: 23 May 2015

Abstract One of the problems in the field of data mining with evolutionary algorithms is the variance of accuracy in multiple runs. Decreasing the variance of accuracy without any accuracy reduction is very difficult since there is a trade-off between these conflicting objectives. In this paper we follow two abstract objectives: accuracy and interpretability. The interpretability is measured by three criteria: number of the rules, sum of the rules lengths and the standard deviation of the accuracy (Acc.SD). The proposed method consists of two stages, and in both, an innovative binary version of the krill herd algorithm has been introduced. In this study, choosing the best krill in population and indicating the local best of the krills in each generation are performed according to a new multi-objective function. In the first stage, candidate rules are generated intelligently using Pittsburgh and iterative rule learning approaches that guarantee the diversity of the extracted rules. The Sifter approach, that is presented here, uses a clustering concept and is incorporated in stage two for robust rule set selection from the candidate rules. Multiple executions of Sifter give roughly the same results. Also in this study, we offer the rule set distance measure that is calculated in two modes: Morphologically and Semantically. Experimental results show that we have successfully improved the two objectives that are naturally in conflict.

Communicated by V. Loia.

✉ Mohammad Saniee Abadeh
saniee@modares.ac.ir

Ali Mohammadi Shanghooshabad
ali.mohammadi@modares.ac.ir

1 Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran

Keywords Sifter approach · Binary krill herd algorithm · Evolutionary fuzzy rule learning · Pittsburgh-iterative rule learning · Rule set distance (RSD) measure

1 Introduction

Because of the importance of extracting the high accurate and high interpretable rules in data mining, many studies have been done in this area (Castro et al. 2007; Liu et al. 2013; De Falco 2013; Nguyen et al. 2013; Zhang et al. 2014). Further studies on rule extraction context use evolutionary algorithms (Nguyen et al. 2013; Kromer et al. 2013; Khalili-Damghani et al. 2013; Hoffmann 2004; Del Jesus et al. 2004; Sánchez et al. 2009; Cordón and Herrera 2001). Some of the evolutionary algorithms follow multiple objectives in rule mining process (Wang et al. 2005; Ishibuchi and Yamamoto 2004; Alcalá et al. 2007; Gacto et al. 2013; Ishibuchi et al. 1997; Ishibuchi and Nojima 2007; Alcalá et al. 2009), they are called Multi-objective evolutionary algorithms (MOEA) and they also can generate fuzzy rule-based systems (FRBSs) to improve interpretability. Many fuzzy algorithms are proposed in order to enhance the interpretability of the results. These algorithms have been employed for various types of applications (Vasilakos et al. 1998; Zikidis and Vasilakos 1996; Pedryez and Vasilakos 1999).

In Ishibuchi et al. (1994) a two-stage algorithm is used, first candidate rules are generated and then the best rule set is selected using a genetic algorithm. In Ishibuchi et al. (1994), two objectives are considered: maximizing the accuracy and minimizing the NR.

In the Ishibuchi et al. (2001), two genetic-based approaches are suggested for generating non-dominated rule sets with respect to three objectives: maximizing the accuracy and minimizing the NR and SRL. The main task in Ishibuchi et al. (2001) is to design interpretable fuzzy rule-based systems with high accuracy.

In Ishibuchi and Yamamoto (2004), the candidate fuzzy rules are generated and prescreened by two rule evaluation measures: confidence and support. So a small number of fuzzy rules are selected from the prescreened rules with a multi-objective evolutionary algorithm. In the rule selection stage, three objectives are considered: maximization of the accuracy, minimization of the NR, and minimization of the SRL.

A genetics-based machine learning (GBML) algorithm is used in Ishibuchi and Nojima (2007) so that the Michigan and Pittsburgh approaches are used in combination. Suggested method in this study pursues three objectives as mentioned in Ishibuchi and Yamamoto (2004).

As it can be seen, most of works that have been done in the fuzzy rule discovery area used the minimizing NR and SRL as interpretability, and according to our knowledge, none of them have not considered the variance of the accuracy in consecutive runs as an interpretability measure.

It seems that if a GBML algorithm achieves high accuracy with high variance, its output would not be trustable; therefore, in this study the main goal is the extraction of high quality and also robust fuzzy rule set. Here, the term robust means that the final rule learning algorithm should be capable to extracting same fuzzy rules in in various execution times.

In this paper, our approach is divided into two stages. In stage one, the candidate rules are generated intelligently. A combination of the two approaches, the iterative rule mining and the Pittsburgh approach, is used to generate the candidate rules. In the second stage, we will search for the best rule set from the candidate rules that follows our two objectives using an imitation-based evolutionary algorithm. To choose the best rule set that satisfies our objectives we have used clustering concept in the second stage. In stage 2, Sifter approach identifies rules granules and attempts to choose the best rules from these granules in which our objectives are satisfied. The Sifter approach works like granular computing methods (Yao et al. 2013). Our objectives are maximizing the number of patterns correctly classified (accuracy), improving the interpretability that is measured with the number of rules (NR), sum of the rules lengths (SRL) and standard deviation of accuracy in consecutive runs (Acc.SD). Calculating the standard deviation measure will be described in the Sect. 4.

The paper is structured as follows: the next section includes a preliminary to *fuzzy reasoning method*. Section 3 describes the paper proposed method. In the Sect. 4, the experimental results are shown and in the last section, the conclusion of the paper is discussed.

2 Preliminary: fuzzy reasoning method

For n-dimensional pattern classification problems, the following fuzzy if–then rules are applied in the design of our robust algorithm:

$$\begin{aligned} R_j : &\text{if} \quad X_1 = A_{j1} \text{ and } X_2 = A_{j2} \text{ and } \ldots \text{ and } X_n = A_{jn} \\ &\text{then class is} \quad C_j \quad \text{with } \mathrm{CF}_j, \quad j = 1, 2, \ldots, N \end{aligned}$$

where R_j is rule index, X_i denotes a linguistic variable, A_{ji} is a linguistic fuzzy term, c is number of classes and $C_j = 1, 2, h$ demonstrate consequent class and CF_j is a certainty grade of this rule in the unit interval [0, 1], and N is a number of rules. All attributes of datasets are converted to numeric attributes, so are normalized between [0, 1].

To improve interpretability of fuzzy rules we used linguistic variables in this study. Variable X_i has a linguistic set $U = \{VL, L, M, H, VH\}$; each value of X_i uniformly shows 1/5 of domain [0, 1]. We also used *don't care* value as all of those values. The total number of if–then rules with n features is 6^n and we aim to construct a rule set by selecting a subset of these rules. This rule set construction process is very computationally expensive since, we should simultaneously dominate two interrelated problems. First learning a pool of high quality fuzzy if–then rules and second selecting a set of cooperative rules as the final fuzzy rule set.

Our work is a c-class problem in the n-dimensional space with numeric attributes. Also we have m real instances $X_p = (X_{p1}, X_{p2}, \ldots, X_{pn})$, $p = 1, 2, 3, \ldots m$ are given as training patterns. In this classification system, the result C_j and certainty grade of rule CF_j will be calculated by the following: first we calculate the compatibility of each training pattern with the fuzzy if–then rule R_j by the following formula then we calculate the whole compatibility grades for each class:

$$\begin{aligned} \beta_{\text{class}\,h}(R_j) &= \sum_{x_p \varepsilon \text{class}\,h} \prod_{i=1}^{i=n} \mu_{A_{ji}}(x_p), \\ p &= 1, 2, \ldots m, \quad h = 1, 2, \ldots, c \end{aligned} \tag{1}$$

where μ_{R_j} is the membership function of A_{ji}. In next step find the consequent class C_j that has maximum $\beta_{\text{class}\,h}(R_j)$:

$$\begin{aligned} \beta_{\text{class}\,C_j} = \max \{ &\beta_{\text{class}\,1}(R_j), \beta_{\text{class}\,2}(R_j), \\ &\ldots, \beta_{\text{class}\,h}(R_j) \} \end{aligned} \tag{2}$$

If there is more than one class with maximum value for $\beta_{\text{class}\,h}(R_j)$, we do not assign any consequent class for R_j. Then we can calculate certainty grade of R_j and for an input

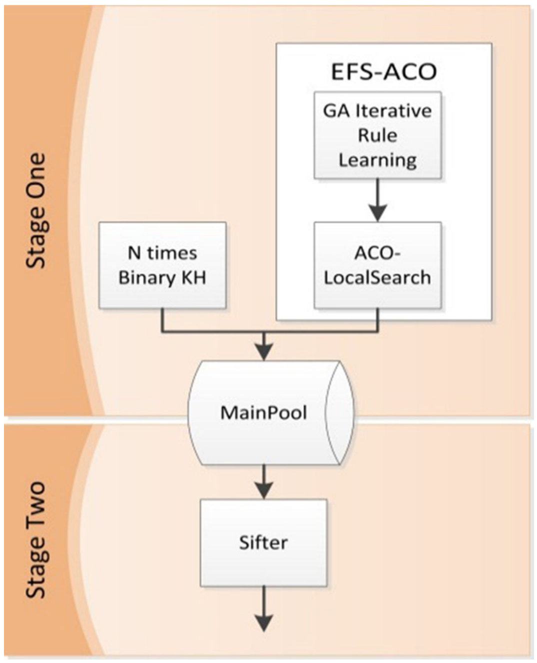

Fig. 1 General structure of the proposed method

pattern $X_p = (X_{p1}, X_{p2}, \ldots, X_{pn})$ the winner rule R_j will be calculated by following (3):

$$\mu_{j^*}(x_p) \cdot \mathrm{CF}_j = \max \left\{ \mu_{j^*}(x_p) \cdot \frac{\beta_{\mathrm{class}\, C_j} - \left(\sum_{h \neq c_j} \beta_{\mathrm{class}\, h}(R_j)/c-1\right)}{\sum_{h=1}^{c} \beta_{\mathrm{class}\, h}(R_j)} : j = 0, 1, \ldots, N \right\} \tag{3}$$

Thus, we can identify the consequent class of input pattern with the aforementioned formula.

3 Proposed method

General structure of the proposed method is shown in the Fig. 1. This method consists of two stages. In the stage one, the candidate rules are generated intelligently and in the stage two, using the Sifter approach, the best rule set from the candidate rules is selected in a robust way. In fact, each run of the Sifter approach will yield the same output.

In the first stage, Pittsburgh-based binary KH is executed several times in a parallel way and output rule sets of them are accumulated in the main pool. So with GA-based iterative rule learning approach, a large number of the rules are generated iteratively and incrementally. Generation of the rules will continue as long as no more record that is not covered with rules remains. The output rules of iterative rule learning are sent to the ACO-Local Search process. The ACO-Local Search process tries to establish a high level relationship between iteratively generated rules. Rules are changed in a way that the relationship between rules increases, leading to a better accuracy for the rule set. Here, the relationship between the rules in a rule set means cooperation among them. Rules in a rule set should cooperate with one another in order to increase the total accuracy of the rule set. In other words, ACO-based local search might weaken a fuzzy rule to strengthen the extracted fuzzy rule set of Genetic-Based Iterative Rule Learning stage. The output rules of the ACO-Local Search are accumulated in the main pool too. It is important to note that in stage one of the proposed method, two fuzzy rule learning algorithms are employed with different and complement nature to fill the main pool. The first algorithm, which is called binary KH, provides general fuzzy if–then rules with high number of *Dont Cares* in antecedent part of each rule. These rules are easy to interpret but suffer from low precision. The algorithm which is called EFS-ACO (this algorithm will be explained in detail in Sect. 3.1) consists of two stages. The first stage of this algorithm learns accurate rules with low number of the *Dont Cares* in antecedent part of the each rule. Although these rules are hard to interpret, they have high accuracy. In fact, the binary KH has high exploration and the EFS-ACO has high exploitation capability. The combination of the learned rules from the two learning algorithms increases the diversity of the if–then rules in the main pool. This high diversity enables the sifter in stage two of proposed method to learn more robust fuzzy rule set, since the sifter has access to nearly every possible high accuracy and high interpretable fuzzy rules.

In the stage two of the paper proposed method, a best rule set is selected from the main pool with using binary KH that is implemented on the Sifter approach. This selection is done in such a way that gives the same answer every time. In this section, the proposed method is described in four subsections. In the first subsection, GA-based iterative rule learning algorithm is explained. In the second subsection, the ACO-Local Search method is described. In the third subsection, the binary-KH algorithm is explained and in the last subsection, our Sifter approach is described.

3.1 GA-based iterative rule learning

The first stage of EFS-ACO is an evolutionary fuzzy system that generates fuzzy if–then rules in an incremental way; this evolutionary algorithm optimizes one of the fuzzy rules at a time. The learning mechanism decreases the weight of training instances that are correctly classified by the optimized rule. Then, the next rule learning cycle focuses on fuzzy rules that are uncovered or misclassified instances currently. In the each iteration, the fuzzy rule that can classify the current dis-

```
Function GA − Iterative(problem) returns a rule set that is generated itarativly
Input: $N_{pop}, P_{crossover}, P_{mutation}, P_{DC}, P_r$
OutPut: Rule Set

population ← ∅
rs ← ∅
while(OuterCycleTeminationTest())do
    label1: Initialization(population)
    while(InnerCycleTeminationTest()) do
        GA − Generation()
        Replacement()
    End While
    rs ← BestRuleOf(Population)
    Weight Adjustment()
End While
return rs
```

Fig. 2 Pseudocode of genetic-based iterative rule learning

tribution of training instances better than other rules of the population is selected to be included in the final fuzzy rule base. Each step of GA-Iterative algorithm is described in Fig. 2.

Initialization The number of fuzzy if–then rules in the population is denoted by N_{pop}. To produce an initial population, N_{pop} fuzzy if–then rules are generated according to a random instance in the train dataset. The proposed evolutionary fuzzy system is considered for each of the classes of the problem separately. Therefore, the mentioned random pattern is selected according to the instances of the training dataset, the consequent class of which is the same as the class that the algorithm works on. The probability for each training instance to be chosen in this step is proportional to its current weight. This means, the algorithm considers a greater probability for those instances that have not been learned in previous iterations. Next, for this random instance, we specify the most compatible combination of antecedent fuzzy sets using only the five linguistic values $U = \{VL, L, M, H, VH\}$. After learning the fuzzy if–then rules, the fitness value of each rule is evaluated by classifying all the given training instances using the set of fuzzy if–then rules in the population.

GA-generation From the current population a pair of fuzzy if–then rules is selected to generate new fuzzy if–then rules for the next population. This selection is performed using the tournament selection strategy. The selection process is iterated until a pre-specified number of pairs of fuzzy if–then rules are selected. A crossover operation is then applied to a selected random pair of fuzzy if–then rules with a crossover probability (an input parameter). We have applied the uniform crossover. With mutation probability (is another input parameter), each dimension of fuzzy if–then rules is randomly replaced with a different antecedent fuzzy set. In this paper, the probability of changing to don't care value is more than the other linguistic values. We call this probability P_{DC}. After selection process, crossover and mutation steps, the fitness value of the generated individuals is evaluated according to Eq. (4).

$$\text{fitness} = \frac{\text{NCP}_i^2}{\text{NMP}_i} \tag{4}$$

where NCP is the number of patterns that correctly classified and NMP represents the number of the misclassified patterns. Suppose we have two krills. If both of them have equal values for $\frac{\text{NCP}_i}{\text{NMP}_i}$, the algorithm focuses on the krill with higher NCP when we use Eq. (4). For example suppose we have a krill with $\text{NCP} = 8$, $\text{NMP} = 4$ and a krill with $\text{NCP} = 4$, $\text{NMP} = 2$; both of the krills have 0.5 values for $\frac{\text{NCP}_i}{\text{NMP}_i}$ but when we use Eq. (4), the fitness of the first krill is better than second krill. Since we are looking up the rules with high NCPs, the Eq. (4) gives better fitness values for the krills with higher NCPs.

Replacement A pre-specified number of rules in the population are replaced with the newly generated rules. Then, P_r percentages of the worst rules with the lowest fitness values are eliminated from the current population and $(100 - P_r)$ percentages of the newly generated rules are added. After the mentioned replacement procedure, the fitness value of the individuals is evaluated according to Eq. (4).

InnerCycleTeminationTest Any stopping condition for terminating the inner cycle of the Iterative-based fuzzy rule learning can be used. In the computer implementation of this paper, the total number of generations is used as a stopping condition.

OuterCycleTeminationTest After termination of the inner cycle of EFS-ACO, the algorithm adds the best fuzzy rule of the evolved population to the final classification rules list and checks if this added fuzzy rule is capable of improving the classification rate of final classification system. If the

```
Function ACO − LocalSearch( ) return improved rule set
Input: $Primary_{rs}[], M, S$
Output: $improved_{rs}$

$SortPrimary_{rs}()$
$t = 1, b = 0$
While $\left( |b - t| <> SizeOf(Primary_{rs}) \right)$ {
    $NewRule \leftarrow PerformACO(Primary_{rs}[t])$
    If NewRule is accepted Then
        $b = t$
        Continue with While * if local search is succesfull then the search process is reapeted on $Primary_{rs}[t]$ *
    $t = t + 1$
    If $t > SizeOf(Primary_{rs})$ Then $t = 1$
}
Return $Primary_{rs}[]$
```

Fig. 3 Pseudocode of ACO-based local search

classification rate is not improved, the algorithm removes the added fuzzy rule from the final classification rules list and terminates. Otherwise, it goes to the next step.

Weight adjustment In the each iteration of the main evolutionary process, rule R_t with the best fitness value is inserted into the primary fuzzy rule base. After each rule generating process, the records that are misclassified will end up having the same weight. The weight of those records that are classified correctly will be became zero. After this step, the algorithm jumps to *Initialization*.

3.2 ACO-local search process

In order to improve the accuracy of the resulted rule set ($\text{Primary}_{\text{rs}}$) from the first stage of EFS-ACO, a secondary stage is used. In the second stage, an ant colony optimization algorithm is applied on the $\text{Primary}_{\text{rs}}$ from the first stage of EFS-ACO. This secondary stage of EFS-ACO is performed as a local search process in which, the ACO improves one rule from the $\text{Primary}_{\text{rs}}$ at a time. The improvement is done by modifying at most M antecedents of the candidate rule from the $\text{Primary}_{\text{rs}} \cdot M$ specifies the maximum allowed possible modifications.

The operation of *ACO-LocalSearch* process in Fig. 3 searches the neighborhood of the current rule and finds a proper modification for it. That is as follows: The *PerformACO* local searcher uses a population of ants to carry out its local search process. This population will searches the neighborhood of the candidate rule and improves it according to the best discovered modification. We are using the ACO as a local search procedure and the representation of ants pheromone paths must be like to a form that can show an instruction of how to change the rule to improve the total accuracy of the $\text{Primary}_{\text{rs}}$. Hence, each path is a string of characters that shows parts of the rule that should be modified. The above discussion implies that the evaluation of each path is done according to the amount of improvement of the $\text{Primary}_{\text{rs}}$ accuracy. The detail of *PerformACO* procedure is as follows: First, initiate the pheromone paths and parameters (e.g. iteration = 0). The value for the initial pheromone is considered according to (5):

$$\tau_{ij}(t = 0) = \frac{1}{\sum_{i=1}^{a} b_i} \tag{5}$$

where a is the total number of attributes (e.g. $a = 41$) and b_i is the number of possible values that can be taken on by attribute A_j (i stands for the attribute index and j is the membership function index). According to Sect. 2, the value of b_i is 6 for all of the attributes. Each ant in the ACO-based local search process generates sequence of the modification according to M. The desirability of each ant for changing the ith antecedent of the candidate rule R_C to A_j is given by (6).

$$p_a(R_C, i, A_j) = \frac{\left[\tau(R_C, i, A_j)\right]\left[\eta(R_C, i, A_j)\right]}{\sum_{u=0}^{5}\left[\tau(R_C, i, A_u)\right]\ \left[\eta(R_C, i, A_u)\right]} \tag{6}$$

In the Eq. (6) τ is the pheromone and η is a heuristic probability, which is 0.5 for $A_j = \text{DC}$ and 0.1 for each of the other linguistic values. This value setting for η enables the ACO-based local searcher to increase the interpretability of R_C while improving the cooperative property of R_C. Therefore, the tendency of ants to generate "*don't care*" is high specifically in the first iterations of the local search process. When ants in the colony generate a modification, the fitness of each modification is calculated and if a better modification is found, the best so found modification in the local search process is updated. The fitness of the modifications is calculated according to the improvement of the classification rate of the original rule set. If the accuracy of the original rule set decreases then the fitness value would be negative. In this situation, the algorithm applies a zero value for the fitness. Increasing the value of pheromone of used antecedent values is according to (7).

$$\tau(R_C, i, A_j) = \tau(R_C, i, A_j) + \tau(R_C, i, A_j) \cdot f(R_C, i, A_j) \tag{7}$$

where $f(R_C, i, A_j)$ denotes the fitness of modification that changes i^h antecedent part of R_C to A_j.

Then, decreasing the value of unused antecedent pheromones to simulate pheromone evaporation in real ant colonies occurs. In the ACO-based local search process, pheromone evaporation is implemented in a somewhat indirect way. In fact, the effect of pheromone evaporation for unused terms is achieved by normalizing the value of each pheromone $\tau(R_C, i, A_j)$. This normalization is carried out by dividing the value of each $\tau(R_C, i, A_j)$ by the summation of all $\tau(R_C, i, A_j), \forall i, j$. If the maximum number of iterations is reached then return the best modification and terminate the local search procedure.

The main point of ACO-based local search in the proposed Sifter is to improve the quality of each learned fuzzy if–then from the previous stage. Since extraction of fuzzy rules in Genetic-Based Iterative Rule Learning stage performs without considering cooperation among rules, the final fuzzy rule set at the end of this stage suffers from low accuracy. The ACO-based local search covers this issue by modifying each rule in the extracted fuzzy rule set from the previous stage. In this modification, the total accuracy of fuzzy rule set is considered rather than improvement of a single fuzzy rule.

3.3 Binary-KH rule mining algorithm

In this subsection Binary KH algorithm is explained and the main four points of binary KH in compared with the krill herd optimization (KH) algorithm (Gandomi and Alavi 2012) and other algorithm are expressed. The KH algorithm has a high exploration and high exploitation ability. One of the advantages of the KH is that we can easily define the sensitivity of algorithm to more exploration or more exploitation with parameters. The KH algorithm is highly efficient, but has a large number of parameters which are considered to be disadvantageous. The KH algorithm has been previously used in continuous optimization problems. Here, we offer a discrete version of KH and also, will show how this algorithm is applied to data mining problems. The core of Binary KH is the same KH algorithm (Gandomi and Alavi 2012). One of the four points is that Binary KH is a binary version of the KH algorithm that has not been studied previously. The second point is that fitness function of the Binary KH is a heuristic function. The third point is that choosing the best krill and local best of the krills in each generation are performed according to a new multi-objective function. The last point is that we have added a new local search process in the algorithm that helps to achieve our objectives. Other parts of the algorithm are like the KH. Pseudocode of the Binary KH is shown in Fig. 4.

The $RandPoolFilling$ function in Fig. 4 is used to generating random rules to fill the random pool and the algorithm tries to intelligently select best combination of random rules. The $RandPoolFilling$ function has two parameters; the RandomPool$_{\text{size}}$ parameter is to determine the capacity of the random pool and DC_Rate parameter is to determine the rate of Dont_Care term in each dimension of rules. The krills of Binary KH are binary and the dimension of each krill is equal to the number of random rules. Each gene of krill indicates the presence or absence of a rule in the random pool. The Fig. 5 is illustrated to show this matter.

The $EvaluatePopulation$ function has a duty to evaluate the fitness of the krills. In this paper we define the fitness as Eq. (4).

Then the $RuleSetAnalyzer$ function eliminates two types of rules. The first type are those in which their NCP is lower than $MinNCP$ (which is a function parameter of $RuleSetAnalyzer$). This $MinNCP$ is based on the validation dataset, and the number of records in the validation dataset is defined as $Validation_{size}$ (another parameter of function). The second type of rules comprises $DelPercent$ (another parameter of function) of the worst rules that must be eliminated. $DelPercent$ represents the percentage of the worst rules in each krill. With the $RuleSetAnalyzer$ function, test accuracy of our algorithm in most of times is higher than the train accuracy, but not always. The Binary KH gradually increases the number of rules in each krill to improve the accuracy and the duty of the $RuleSetAnalyzer$ function is to decrease the number of rules while holding the accuracy high. This process is illustrated in Fig. 6. The upper line in Fig. 6 shows the NCP of the best krill in population and the lower line shows the NR of the best krill. The horizontal axis shows the generations. As you can see the algorithm tries to improve NCP with increasing the NR and then tries to decrease NR with holding high NCP.

Recognition of the best krill in population and indication of the local best of a krill are done as follows. Suppose we have two krills. Both have the fitness values and NR. The krill that has the best fitness will be selected and if the both have same fitness then the krill with less NR will be selected and if they have same NR values then the krill with lower SRL will be selected.

The general policy of the KH is that krills are moving toward food and spaces with more congestion. In their natural system, the fitness of each individual is supposed to be a combination of the distance from the food and from the highest density of the krill swarm (Gandomi and Alavi 2012). The

```
Function Binary KH(problem) returns a state that is a local maximum
Input: $Population_{size}, Problam_{size}, RandomPool_{size}, \omega_n, N^{max}, D^{max}, \omega_f, \Delta t, P_{crossover}, P_{mutation}, Validation_{size}$, MinNCP, $DelPercent, DC_{Rate}$
Output : $RS_{best}$
$RndPool \leftarrow RandPoolFilling(RandomPool_{size}, DC_{Rate})$
$Population \leftarrow \emptyset$
$P_{best} \leftarrow \emptyset$
For i = 1 to $Population_{size}$ do
    $K_iN \leftarrow RandomVelocity()$
    $K_iF \leftarrow RandomVelocity()$
    $K_iPosition \leftarrow RandomPsition(Problam_{size})$
    $K_ilbest \leftarrow K_iPosition$
    Population $\leftarrow K_i$
End For
$I \leftarrow 0$
while $\neg$ StopCondition() do
    EvaluatePopulation(Population)
    RuleSetAnalyzer(Population, $Validation_{size}, MinNCP, DelPercent$)
    $K_b = GetBestSolution(population)$
    $K_w = GetWorstSolution(population)$
    For each $K_i \in Population$ do
        $K_iN \leftarrow UpdateNIntention(population, Population_{size}, K_iN, K_b, K_w, \omega_n, N^{max}, I)$
        $K_iF \leftarrow UpdateFIntention(population, Population_{size}, K_iF, K_ilbest, \omega_f, N^{max}, I)$
        $K_iD \leftarrow RandomIntention(D^{max}, I)$
        $K_ivelocity \leftarrow$ UpdateVelocity ($K_iN, K_iF, K_iD$)
        $K_iposition \leftarrow$ UpdatePosition ($K_iposition, K_ivelocity, \Delta t$)
        $K_iposition \leftarrow Recombination$(Population, $K_iposition, K_b, P_{crossover}$)
        $K_iposition \leftarrow Mutation$(Population, $K_iposition, K_b, P_{mutation}$)
        If $Fitness(K_iposition)$
                    $\geq Fitness(K_ilbest)$ or $(Fitness(K_iposition) = Fitness(K_ilbest)$ and $NumOfRules(K_iposition)$
                    $\leq NumOfRules(K_ilbest))$ or $(Fitness(K_iposition) = Fitness(K_ilbest)$ and $NumOfRules(K_iposition)$
                    $= NumOfRules(K_ilbest)$ and $SRL(K_iposition) \leq SRL(K_ilbest))$ Then
            $K_ilbest \leftarrow K_iposition$
        End If
    End For each
    $I \leftarrow I + 1$
End While
EvaluatePopulation(Population)
$S_{best} \leftarrow GetBestSolution(population)$
return $RS_{best}$
```

Fig. 4 Pseudocode Of binary KH

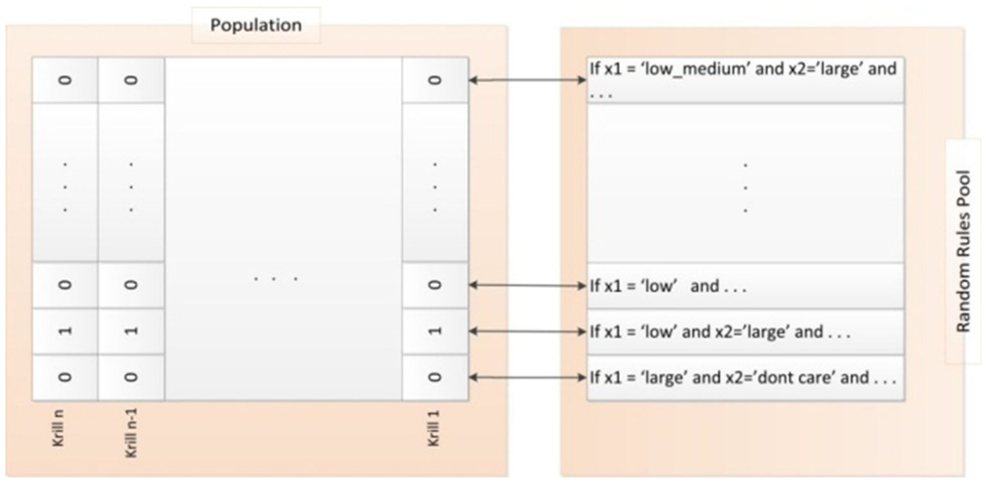
Population
0
1
Krill n
Krill n-1
Krill 1
If x1 = 'low_medium' and x2='large' and ...
If x1 = 'low' and ...
If x1 = 'low' and x2='large' and ...
If x1 = 'large' and x2='dont care' and ...
Random Rules Pool

Fig. 5 Representation of krills

time-dependent position of an individual krill in 2D surface is explained by the following three main actions (Hofmann et al. 2004): for ith krill we have

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{8}$$

where N_i is the motion induced by other krill individuals that is calculated by $UpdateNIntention()$ function in Fig. 4; F_i is the foraging motion that is calculated by $UpdateFIntention()$ function in Fig. 4, and D_i is the physical diffusion of the ith krill individual that is calculated by

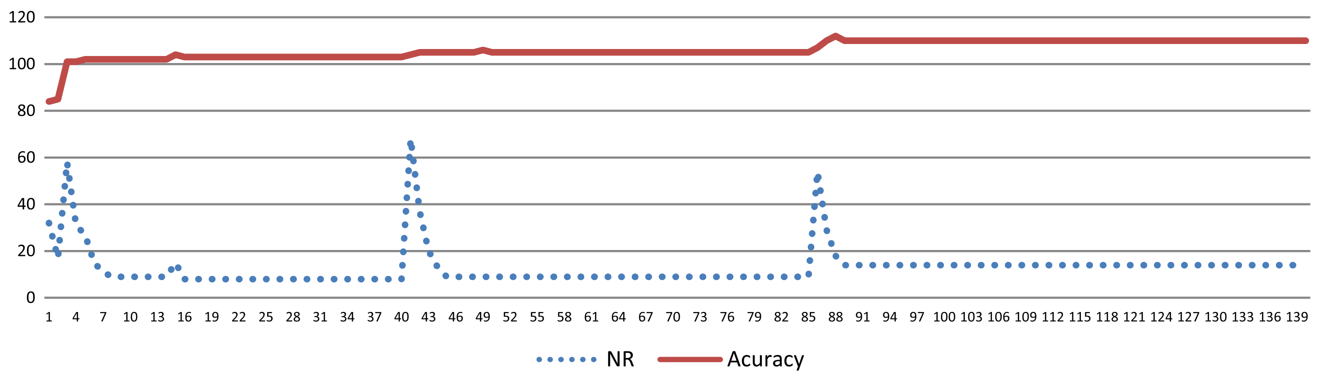

Fig. 6 Execution of binary KH on the heart dataset

RandomIntention() function in Fig. 4; these are explained in Gandomi and Alavi (2012) but we summarize it here.

Movement induced by other krills (N) The krill tries to keep a high density and move due to their mutual effects (Hofmann et al. 2004). The direction of motion induced is estimated from the local effect of swarm density, a target effect of the individual swarm density, and a repulsive effect (Hofmann et al. 2004).

$$N_i^{\text{new}} = N^{\max} \cdot \alpha_i + \omega_n \cdot N_i^{\text{old}} \tag{9}$$

$$\alpha_i = \alpha_i^{\text{local}} + \alpha_i^{\text{target}} \tag{10}$$

$$\alpha_i^{\text{local}} = \sum_{j=1}^{NN} \hat{K}_{ij} \cdot \hat{X}_{ij} \tag{11}$$

$$\hat{X}_{ij} = \frac{X_j - X_i}{\|X_j - X_i\| + \varepsilon} \tag{12}$$

$$\hat{K}_{ij} = \frac{K_j - K_i}{K^{\text{worst}} - K^{\text{best}}} \tag{13}$$

$$\alpha_i^{\text{target}} = C^{\text{best}} \hat{K}_{i,\text{best}} \cdot \hat{X}_{i,\text{best}} \tag{14}$$

Here, ω_n is the inertia weight of the motion induced in the range [0, 1], $N^{\max}$ is the maximum induced speed, according to the estimated values of the maximum induced speed (Hofmann et al. 2004), it is taken like $0.01\,\text{ms}^{-1}$. N_i^{old} is the last motion induced, α_i^{local} is the local effect by the neighbors and α_i^{target} is the target direction effect by the best individual. The effect of the krill neighbors can be supposed as an attractive tendency and repulsive tendency between the krills for a search in local region. K^{best} is the best and K^{worst} is the worst fitness values of the individuals; K_i shows the fitness value or the objective function value of the ith individual; K_j is the fitness value of jth ($j = 1, 2, \ldots, NN$) krill neighbor; X is used for the related krill positions; and NN is used for the number of the krill neighbors. For shirking the singularities, a small value ε is added to the denominator. C^{best} is used for the effective coefficient of the individual with the best fitness to the ith individual.

Foraging activity shown with (F) First, we define a virtual food (will be explained later) and then we calculate two main effective parameters as the foraging motion. First, the food position is calculated and then the previous experience about the food position will be calculated (Gandomi and Alavi 2012).

$$F_i = V_f \cdot \beta_i + \omega_f \cdot F_i^{\text{old}} \tag{15}$$

$$\beta_i = \beta_i^{\text{food}} + \beta_i^{\text{best}} \tag{16}$$

where, V_f is used for the foraging speed, ω_f is used for the inertia weight of the foraging motion in the range [0, 1], F_i^{old} is used for the last foraging motion, β_i^{food} is used for the food attraction and β_i^{best} is used for the effect of the best fitness of the ith krill. According to the estimated Quantities for the foraging speed (Price 1989), it is taken as $0.02\,\text{ms}^{-1}$.

$$X^{\text{food}} = \frac{\sum_{i=1}^{N} \frac{1}{K_i} \cdot X_i}{\sum_{i=1}^{N} \frac{1}{K_i}}, \quad \text{for minimization} \tag{17}$$

$$\beta_i^{\text{food}} = C^{\text{food}} \cdot \hat{K}_{i,\text{food}} \cdot \hat{X}_{i,\text{food}} \tag{18}$$

$$\beta_i^{\text{best}} = \hat{K}_{i,\text{best}} \cdot \hat{X}_{i,\text{best}} \tag{19}$$

Here, C^{food} is the food coefficient. First, the center of food should be found using the Eq. (17) and in the second step, try to formulate food attraction using the Eq. (16). It should be noted that this attraction quantity not be accurately calculated; it is estimated.

Random diffusion shown with (D) krill physical diffusion is considered a random process (Gandomi and Alavi 2012). This motion can be express as a maximum diffusion speed and a vector with random directions. It is calculated as follows:

$$D_i = D^{\max} \cdot \left(1 - \frac{I}{I_{\max}}\right) \cdot \delta \tag{20}$$

Here, $D^{\max} \in [0.002, 0.01]\,(\text{ms}^{-1})$ and δ is used for the random directional vector. Its array has random values between

```
**Function** $Sifter()$ returns a state that is a local maximum
**Input**: $MainPool$, QualifiedPool$_{size}$, $Sieve_{size}$, $SelectionType$, $Validation_{size}$, MinNCP, $DelPercent$
**Output** : $FRBS$

$SortAscending(MainPool)$
**For** $i = 1$ to $\frac{\text{QualifiedPool}_{size}}{4}$ **do**
    $QualifiedPool \leftarrow GetRule(MainPool, i)$
    remove selected rules from MainPool
**End For**
$QualifiedPool \leftarrow selection\left(MainPool, QualifiedPool, SelectionType, \frac{3*\text{QualifiedPool}_{size}}{4}\right)$
$size \leftarrow SizeOf(QualifiedPool)$
**While**$(i < Sieve_{size}$ and $i < Size)$
    $Sieve \leftarrow GetRule(QualifiedPool, i)$
    remove ith rule from QualifiedPool
    $i \leftarrow i + 1$
**End While**
$FRBS \leftarrow \emptyset$
**Do**
    $size = SizeOf(QualityPool)$
    $FRBS \leftarrow KH_{Sifter}(Sieve, FRBS)$
    $Sieve \leftarrow \emptyset$
    **IF** $size > Sieve_{size}$ **Then**
        $Sieve \leftarrow Selection(QualifiedPool, FRBS, SelectionType, Sieve_{size})$
    **Else**
        $Sieve \leftarrow Selection(QualifiedPool, FRBS, SelectionType, size)$
    **End IF**
**while** $(size > 0)$
**return** $FRBS$
```

Fig. 7 Pseudocode of Sifter approach

-1 and 1, I is ith iteration of algorithm and $I_{\max}$ is used to show maximum iteration for algorithm. The Lagrangian model is generalized to an n-dimensional decision space in krill herd algorithm.

$$X_i(t + \Delta t) = X_i(t) + \Delta t \frac{dX_i}{dt} \tag{21}$$

$$\Delta t = C_t \sum_{j=1}^{\text{NV}} \text{UB}_j - \text{LB}_j \tag{22}$$

where NV is used for the total number of variables, and LB_j and UB_j are lower and upper bounds of the jth variables ($j = 1, 2, \ldots, \text{NV}$), respectively. C_t is a constant number between [0, 2].

The major difference between binary KH with continuous version is that $X_i(t + \Delta t)$ is normalized by sigmoid function, then a random number between [0.5, 1] (that is experimentally obtained) is generated; if sigmoid $[X_i(t + \Delta t)]$ is higher than random number, 1, and else 0 is applied. The output of the binary-KH in the stage one is a rule set that has the high accuracy, low NR and low SRL.

3.4 Sifter approach

Figure 7 represents pseudocode of Sifter approach. To understand the operation of Sifter Approach, suppose we have a sack of objects. Sifting process is used for a robust selection of the objects, so that in each sifting, the same objects are selected. First, we fill the sieve with the objects of sack and begin to sift. After sifting the objects that are unwanted, according to the sieve type, the best objects remain in the sieve so we save them elsewhere. Again, the sieve is filled with the sack objects. The Sifter approach used here works like the mentioned process. The rules are the objects that should be refined by the sieve. Here, we use rule pool (RP) as the sack. According to the sifter pseudocode, First QualifiedPool$_{\text{size}}$ (which is an input parameter) number of the serviceability rules are separated and are saved in the $QualifiedPool$. This separation has many benefits. For example, some of the main pool rules are not useful and must be removed from the process. Another advantage of the separation is selection of the rules in different levels.

In this paper, the rules are divided into two different levels, backbone rules and sidebar rules. The backbone rules

```
**Function** Selection( ) Select a rule set from RS1 according to RS2 with the specific SelectionType
**Input**: RS1, RS2, SelectionType, NumberOfNeededRules
**Output** : RS

$RS \leftarrow \emptyset$
**IF** SelectionType = Geno **then**
    $RS \leftarrow GenoSelection(RS1, RS2, NumberOfNeededRules)$
**Else IF** SelectionType = Pheno **then**
    $RS \leftarrow PhenoSelection(RS1, RS2, NumberOfNeededRules)$
**Else IF** SelectionType = Both **then**
    $RS \leftarrow BothSelection\left(RS1, RS2, \frac{NumberOfNeededRules}{2}\right)$
    Remove selected Rules From RS1
    $RS \leftarrow PhenoSelection\left(RS1, RS2, \frac{NumberOfNeededRules}{4}\right)$
    Remove selected Rules From RS1
    $RS \leftarrow GenoSelection\left(RS1, RS2, \frac{NumberOfNeededRules}{4}\right)$
**End IF**
**return** RS
```

Fig. 8 Pseudocode of selecting the rules from a Pool

were stronger than others in stage one and had better fitness values than others. The sidebar rules have a lower impact but they are important anyway. The spaces that are covered by them are not covered by others and this makes them important. Using above-mentioned separation method, best backbone and sidebar rules are selected. In the early steps, sifter approach selects the main and strong rules from the backbone rules while it reaches a steady state. Then it tries to achieve the objectives with using the sidebar rules. Division of the rules is performed by Selection function in sifter approach implicitly.

Shrinking the search space is another advantage of the rules separation. The algorithm will perform better with smaller search space and lower rules. In fact, with the rule separation, we will have comprehensive rules that come from all of the search space which has become smaller with this division. First six lines of the pseudocode correspond to the separation process. After sorting the main pool by fitness, One-fourth of the $QualifiedPool$ is filled with top rules in sorted main pool and three-quarters of QualifiedPool is filled with the Selection function that is shown in next pseudocode. After the separation and filling the $QualifiedPool$, the sift process begins. In the seventh to twelfth lines, $\text{Sieve}_{\text{size}}$ (another input parameter) number of the $QualifiedPool$ rules are picked and are saved in the $Sieve$. The do-while section of the pseudocode, first runs the $\text{KH}_{\text{Sifter}}$ process on $Sieve$ and again, fills the $Sieve$ by Selection function until the $QualifiedPool$ rules are finished and this pool is emptied. The $\text{KH}_{\text{Sifter}}$ process is rather similar to binary-KH, albeit with a little difference. The difference is that here there is a different way to evaluate krills (rule set) and there is no Random Pool. Instead of the random pool, ready candidate rules ($Sieve$ rules that generated in stage one) are sent to the algorithm. The $\text{KH}_{\text{Sifter}}$ function has a new input parameter named FRBS. FRBS is a fuzzy rule set and is the result of the previous run of the $\text{KH}_{\text{Sifter}}$ function. At first, FRBS is empty. The evaluation process of a krill (rule set) is as follows: every time a krill is being evaluated in the $\text{KH}_{\text{Sifter}}$ function, first FRBS is added to the krills rule set and then will be evaluated. In fact, we use the result of the previous step in the current step. It is possible to employ other meta-heuristic search algorithms instead of the binary KH in the structure of the Sifter approach. When the $QualifiedPool$ rules are finished, the last line of the pseudocode returns a fuzzy rule set which is amassed in several steps of sifting.

RS1, RS2 and RS in the Fig. 8 are rule sets. The Selection function, with two input rule sets and $SelectionType$ parameters, tries to select $NumberOfNeededRules$ rules from RS1 by considering RS2 in three different ways: Geno, Pheno and Both. To implement, three functions are needed. The $GenoSelection$ function that is considering RS2, tries to pick the rules from RS1 that are morphologically dissimilar to RS2. This dissimilarity is measured with rule set distance (RSD). RSD is a new measure to calculate dissimilarity content (this is explained in next paragraph). The $phenoSelection$ function, by considering RS2, tries to pick the rules from RS1 that are dissimilar to RS2 semantically. The $BothSelection$ function by considering RS2, tries to pick the rules from RS1 that are dissimilar to RS2 semantically and morphologically. Finally, the Selection function returns the selected rule set.

The $GenoSelection$ function in Fig. 9 returns a rule set from RS1 that have top morphologicallay dissimilar rules to RS2. The $RulesDistance$ function returns a matrix which contains morphologically similarity values of RS1 in comparison with RS2 (rule by rule). Rows of the array represent

```
**Function** GenoSelection( ) returns top dissimilar rules from RS1 compared to RS2 by geno policy (morphologically)
**Input**: RS1, RS2, NumberOfNeededRules
**Output** : RS

$2D_{array} \leftarrow RulesDistance(RS1, RS2)$
Calculate Sum Of Distances in Columns
sorting the RS1 rules according to the columns total distance
RS    ← select top NumberOfNeededRules Rules from ranked rules
**return** RS
```

Fig. 9 Pseudocode of GenoSelection

```
**Function** PhenoSelection(RS1, RS2) returns top dissimilar rules from RS1 in comparison with RS2 by pheno policy(semantical
**Input**: NumberOfNeededRules
**Output** : RS

$i = 0$
**For Each** rule $\in$ RS1 **do**
    $tempRS = RS2$
    $tempRS \leftarrow rule$
    Evaluate (tempRS)
    $array[i] \leftarrow tempRS$
    $i = i + 1$
**End For**
sort array with rule sets fitness
RS ← select top NumberOfNeededRules rules from array
**return** RS
```

Fig. 10 Pseudocode of phenoSelection

the RS2 rules and columns represent the RS1 rules. The similarity values (RSDs) are calculated in the following way: in comparing two rules, if they have the same value (fuzzy term) in feature$_i$, similarity amount will be added with 1. Finally similarity values are divided by length of the rules. After obtaining this matrix, sum of the values in each column are calculated. The columns (RS1 rules) are sorted based on the sum of these values and so, top $NumberOfNeededRules$ rules from ranked rules are passed to Selection function.

The $phenoSelection$ function in Fig. 10 returns a rule set from RS1 that has the top semantically dissimilar rules to RS2. Each of the RS1 rules is separately added to RS2 and evaluated. Then the evaluated rule sets are sorted with rule set fitness. After sorting, the top $NumberOfNeededRules$ rules from the ranked rules are passed to Selection function.

4 Experimental results

In this section, characteristics of datasets and the parameters used in the stage one (generating candidate rules) and stage two (Sifting stage) will be shown. All of the parameter values are calculated experimentally. The results of the runs will be displayed, analyzed and compared with ten other evolutionary fuzzy rule learner algorithms.

4.1 Datasets

The datasets used in the paper are listed in the Table 1. Presence or absence of missing values and the way of the dealing with missing value are also expressed.

4.2 Parameter setting

Some of parameters used in binary-KH and IRL are invariant parameters in our work. These parameters are shown in Table 2. The parameters that are not starred are just for the binary-KH and Sifter but the starred parameters that are for the binary-KH, IRL and Sifter. In Table 2, the first column shows the population size, the second column shows the number of generations, $P_{\text{crossover}}$ shows the probability of crossover operation and P_{mutation} shows the probability of the mutation operation. Validation$_{\text{size}}$ shows number of the validation patterns. The $DelPercent$ parameter is related to $RuleSetAnalyzer$ step of binary KH. Other parameters of binary-KH and IRL algorithms have been described in Sect. 3.

Table 1 Datasets description

	Number of patterns	Number of features	Number of classes	Missing values	Elimination of missing values
Breast (original)	699	9	2	Yes	Average
Liver-disorders	345	6	2	No	None
CMC	1473	9	3	No	None
Heart (stat log)	270	13	2	No	None
Iris	150	4	3	No	None
PIMA (diabetes)	768	8	2	Yes	Average

Table 2 Invariant parameter values of binary-KH and IRL

Population$_{size}$*	NumOfGeneration*	ω_n	N^{max}	D^{max}	ω_f	Δt	$P_{crossover}$*	$P_{mutation}$*	Validation$_{size}$	DelPercent
50	200	0.01	0.01	0.002	0.5	0.5	0.9	0.2	1/5 of train	0.08

* Basic GA parameters

Table 3 Variant parameter values of binary-KH and IRL with number of extracted rules

Parameters	Breast	CMC	Heart	Iris	Liver	PIMA
Binary KH RandomPool$_{size}$	600	600	400	400	400	600
Binary KH number of runs	30	50	30	30	30	50
Binary KH MinNCP	5	5	7	5	6	6
Binary KH Don't Care rate	0.7	0.85	0.8	0.9	0.85	0.85
Max number of rules parameter in IRL	150	600	300	150	300	300
Average main pool sizes in 5 folds	445.2	1129.4	384.4	190	349	603.2

Table 4 Variant parameter values of Sifter

	QualifiedPool$_{size}$	Sieve$_{size}$	MinNCP
Breast	400	30	9
CMC	600	27	25
Heart	300	15	10
Iris	100	20	3
Liver	300	20	10
PIMA	450	40	9

Table 3 shows values of the stage one variant parameters. The four first parameters in Table 3 are related to binary KH that are described in Sect. 3. The sixth row of the Table 3 corresponds to IRL method in stage one of the proposed method. The last row shows the number of extracted rules after the completion of first stage. In fact, the last row shows the average number of extracted rules in each fold.

The variant parameters of Sifter on all data sets are shown in Table 4. The QualifiedPool$_{size}$ determines the size of $QualifiedPool$ and the Sieve$_{size}$ determines the size of $Sieve$. The *MinNCP* specifies the value of *MinNCP* parameter in the Stifer. Invariant parameters of the Sifter are taken from Table 1. The $SelectionType$ value in this paper is $Both$.

4.3 Calculating the measures

A 5 fold validation is implemented. The algorithm is performed 15 times on each fold. Three measures are calculated in each run on the folds: the accuracy, sum of the rule lengths and the number of rules. Results of the 15 runs on Heart data set are shown in Table 5.

The last two rows of Table 5 represent average and standard deviation of measures in 15 runs on the folds. After completing the Table 5, our four measures are calculated. To calculate these objectives, we must calculate average of the averages and averages of the SDs on each of the 5 folds. These four measures on Heart data set are listed in Table 6.

Results for all datasets, after executing our algorithm, are shown in Table 7. Columns of Table 7 show the datasets and the rows represent the accuracy, NR, SRL and the average of the consumed time to execution of 15 runs. Time pattern is "minute: second". Each cell of table contains the average of averages and the averages of standard deviations in 15 runs.

4.4 Analysis of results

In the paper, four measures are considered: improving the accuracy, decreasing the variance of the accuracy, decreasing NR and decreasing SRL.

Table 5 The way of measure calculation

	NR fold 1	SRL fold 1	Acc. fold 1	NR fold 2	SRL fold 2	Acc. fold 2	NR fold 3	SRL fold 3	Acc. fold 3	NR fold 4	SRL fold 4	Acc. fold 4	NR fold 5	SRL fold 5	Acc. fold 5
Run 1	7	11	88.889	6	10	88.889	2	3	83.333	5	13	87.037	7	11	88.889
Run 2	6	9	85.185	5	9	90.741	2	3	83.333	7	14	87.037	7	11	88.889
Run 3	7	11	85.185	6	10	88.889	2	3	83.333	5	13	87.037	7	11	88.889
Run 4	7	11	85.185	6	10	88.889	2	3	83.333	5	13	87.037	7	11	88.889
Run 5	7	11	85.185	6	10	88.889	2	3	83.333	7	14	87.037	7	11	88.889
Run 6	7	11	85.185	5	9	90.741	2	3	83.333	7	14	87.037	7	11	88.889
Run 7	7	11	85.185	6	10	88.889	2	3	83.333	4	9	88.889	7	11	88.889
Run 8	6	9	85.185	5	9	90.741	2	3	83.333	7	14	87.037	7	11	88.889
Run 9	7	11	85.185	5	9	90.741	2	3	83.333	7	14	87.037	7	11	88.889
Run 10	7	11	85.185	6	10	88.889	2	3	83.333	5	13	87.037	7	11	88.889
Run 11	7	11	85.185	5	9	90.741	2	3	83.333	5	13	87.037	7	11	88.889
Run 12	7	11	85.185	5	9	90.741	2	3	83.333	7	14	87.037	7	11	88.889
Run 13	7	11	85.185	6	10	88.889	2	3	83.333	5	13	87.037	7	11	88.889
Run 14	7	11	85.185	6	10	88.889	2	3	83.333	5	13	87.037	7	11	88.889
Run 15	7	11	85.185	5	9	90.741	2	3	83.333	5	13	87.037	7	11	88.889
Average	6.866	10.733	85.432	5.533	9.533	89.753	2.000	3.000	83.333	5.733	13.133	87.160	7.000	11.000	88.889
SD	0.351	0.704	0.956	0.516	0.516	0.956	0.000	0.000	0.000	1.100	1.246	0.478	0.000	0.000	0.000

Table 6 The objectives on the heart data set

5 fold	NR	SRL	Accuracy
Average of the averages	5.427	9.480	86.914
Average of SDs	0.394	0.493	0.478

Our multi-objective polices in the global best selection of the population and indication of the local best of a krill are as follows. Suppose we have two krills. Both have the fitness values and NR. The krill that has the best fitness will be selected and if the both have same fitness then the krill with less NR will be selected and if they have same NR values then the krill with lower SRL will be selected. With this policy, the accuracy, NR and SRL are considered by the algorithm respectively and improve the accuracy and the interpretability of results.

One of the reasons of improvement in accuracy is using the Main Pool. In the second stage of the algorithm (Sifter), only the backbone and sidebar rules compete for being selected. Because only the good rules are in competition, reaching a higher accuracy is very likely. Another reason for improving the accuracy is the clustering logic within the Sifter algorithm. In fact, each time the $Sieve$is is filled, it seems like we separate a cluster of the rules pool. In the Sifter, every time the $Sieve$is being filled, the rules are selected that have less morphologically and semantically similarity to the rules that have already been chosen by the sifter. So, high quality rules are selected initially, and this causes the low quality rules to be excluded from selecting. The $RuleSetAnalyzer$ function does almost the same thing. With eliminating three types of the rules in the krill, only the high quality rules will be allowed to be elected.

The main reason to reduce the standard deviation of the accuracy in consecutive runs is the clustering concept within the sifter algorithm and the duplicated rules in the Main Pool. Working with the clusters) ($Sieves$) makes the algorithm search space smaller and selection will be easier in a smaller space. At any step of the Sifter, when a cluster of rules is selected, search space becomes smaller and KH-Sifter returns rather similar results in different runs. Also, having more duplicated rules increases the probability of choosing the same rules in consecutive runs. Using the Main Pool and the $RuleSetAnalyzer$ functions are the other reasons in reducing the SD of the accuracy. Using the Main Pool and *RuleSetAnalyzer* functions lead to selecting the best rules with high qualities. If the rules of the pool are not changed, and if always the best rules are selected then a robust algorithm is reached that returns the same results in consecutive runs.

Another objective is decreasing the number of rules. The principle reason in reduction of NR is using the $RuleSetAnalyzer$ function. The $RuleSetAnalyzer$ function prunes three types of rules; the first type are the rules that their NCP is lower than *MinNCP* while the krills are evaluated with validation patterns; the second type of rules are $DelPercent$ (an input parameter) percent of the worst rules in each krill; the third type of the rules that must be removed are the duplicated rules in each krill. When you increase the values of *MinNCP* and $DelPercent$, the NR and SDs (variance of the accuracy, NR and SRL) will be reduced. Also, using the Main Pool has an effect on the reduction of NR. If the high quality and gigantic rules are selected in the early steps, selection of large number of low quality rules will rarely occur.

The reduction in the NR is one of the main reasons of decreasing SRL. Lower NR leads to lower SRL. Also, *Don't Care rate* parameter in generating candidate rules (stage one) has a huge influence on reducing SRL of selected rules in the Sifter. When the *Don't Care rate* is high, the SRL of Sifter outputs is low.

When we use the Main Pool, an interesting event occurs in results. Despite the SDs of the NR and SRL being high, the SD of accuracy is low. For example, on breast dataset the SDs of the NR and SRL are 3.338 and 1.037 respectively, but the SD of accuracy is 0.428. When the number of the high quality rules is high, after sifting process the SD of the accuracy becomes low.

4.5 Comparison with others

To compare our results with other studies, we have used KEEL software (Alcalá-Fdez et al. 2009; Derrac et al. 2015). We have downloaded the latest version of the software (V2014-01-29) from http://www.keel.es. Ten evolutionary rule learning algorithms are used in which 5 of them learn

Table 7 Results of Sifter on all datasets

	Breast	CMC	Heart	Iris	Liver	PIMA
Accuracy	97.378 ± 0.428	53.723 ± 0.240	86.914 ± 0.478	97.533 ± 0.000	69.411 ± 0.265	79.456 ± 0.387
NR	16.853 ± 3.338	14.507 ± 1.665	9.480 ± 0.493	5.053 ± 0.570	9.160 ± 0.881	18.573 ± 1.940
SRL	11.067 ± 1.037	8.840 ± 0.674	5.427 ± 0.394	4.707 ± 0.404	5.573 ± 0.525	12.720 ± 0.949
Time avg.	12:29	20:18	9:10	1:28	3:35	22:01

Table 8 Comparison with other methods

	Breast	CMC	Heart	Iris	Liver	PIMA
GFS-GCCL-C (Ishibuchi et al. 1999)	94.248 ± 2.176	45.892 ± 0.726	78.802 ± 5.094	93.022 ± 4.470	63.010 ± 1.213	69.905 ± 1.949
GFS-GP-C (Derrac et al. 2015)	92.915 ± 2.321	49.493 ± 2.540	77.202 ± 0.046	90.756 ± 5.694	60.560 ± 4.327	75.064 ± 2.365
GFS-SP-C (Sánchez et al. 2001)	90.350 ± 2.498	48.647 ± 2.972	80.185 ± 3.618	85.933 ± 7.750	59.652 ± 5.555	74.117 ± 2.756
SGERD (Mansoori et al. 2008)	93.522 ± 0.669	50.046 ± 2.189	79.148 ± 5.103	94.356 ± 2.682	64.720 ± 2.711	74.205 ± 2.346
SLAVE (González and Pérez 2001)	95.320 ± 0.698	42.660 ± 6.249	80.247 ± 3.857	95.956 ± 0.172	59.343 ± 3.066	73.734 ± 1.349
CPSO-C (Liu et al. 2004)	90.901 ± 6.257	46.212 ± 4.063	71.580 ± 7.996	86.978 ± 9.309	61.548 ± 5.731	69.342 ± 4.380
PSO-ACO-C (Sousa et al. 2004)	91.492 ± 3.890	44.545 ± 3.284	66.049 ± 5.666	74.667 ± 10.593	62.807 ± 5.226	71.481 ± 3.106
UCS-C (Bernadó-Mansilla and Garrell-Guiu 2003)	96.062 ± 1.109	46.003 ± 2.374	80.123 ± 3.961	74.667 ± 14.654	62.159 ± 6.409	72.081 ± 3.498
XCS-C (Wilson 1995)	94.65 ± 1.21	47.228 ± 3.788	57.975 ± 5.631	89.778 ± 8.221	63.739 ± 6.149	69.725 ± 3.342
MPLCS-C (Bacardit and Krasnogor 2009)	95.748 ± 1.006	**55.196 ± 1.938**	82.222 ± 3.818	96.044 ± 1.576	65.899 ± 4.691	75.142 ± 1.959
Sifter	**97.378 ± 0.428**	53.723 ± 0.240	**86.914 ± 0.478**	**97.533 ± 0.000**	**69.411 ± 0.265**	**80.956 ± 0.387**

Best result in each column has been bolded crisp rules and 5 of them learn fuzzy rules in an evolutionary way. Experimentally, we have tried to set the best parameters for them. We just compare our results in terms of accuracy and SD of accuracy, because the number of rules (NR) in KEEL algorithms is a fixed parameter that is input. Since we have set it, we do not compare KEEL algorithms with our algorithm in aspects of NR and SRL. To set this parameter, we set it with the mean value of NR in our algorithm. As it can be seen, our two-stage algorithm works better than other algorithms. Our emphasis is that if there is a better rule learning algorithm with results better than our algorithm, if we use it in first stage of our algorithm, not only its results will be improved but also that algorithm will be turned to a robust algorithm. In fact, the stage 2 of our algorithm can be used as a post-processing algorithm on all of the other algorithms. The results are listed in Table 8.

5 Conclusion

In this paper, improving the accuracy and interpretability was our objective. To improve interpretability, three criteria were measured: NR, SRL and SD of the accuracy. To achieve our objectives, the main pool was filled using two rule learning approaches: the Pittsburgh and Iterative. Then a new approach was introduced for robust selection of the best rule set from the main pool; which we named it as Sifter. This paper offered a new division of the rules that helps Sifter work better: the backbone rules, and the sidebar rules. Also in this study, we offered the rule set distance (RSD) measure that is calculated in two modes: morphologically and semantically. Finally, the experimental results showed that we have successfully improved the two objectives that are naturally in conflict.

References

Alcalá R, Gacto MJ, Herrera F, Alcalá-Fdez J (2007) A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. Int J Uncertain Fuzzy Knowl-Based Syst 15:539–557

Alcalá R, Ducange P, Herrera F, Lazzerini B, Marcelloni F (2009) A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems. IEEE Trans Fuzzy Syst 17:1106–1122

Alcalá-Fdez J, Sánchez L, García S, del Jesús MJ, Ventura S, Garrell J, Otero J, Romero C, Bacardit J, Rivas VM (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13:307–318

Bacardit J, Krasnogor N (2009) Performance and efficiency of memetic pittsburgh learning classifier systems. Evol Comput 17:307–342

Bernadó-Mansilla E, Garrell-Guiu JM (2003) Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evol Comput 11:209–238

Castro JL, Flores-Hidalgo L, Mantas CJ, Puche JM (2007) Extraction of fuzzy rules from support vector machines. Fuzzy Sets Syst 158:2057–2077

Cordón O, Herrera F (2001) Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems. Fuzzy Sets Syst 118:235–255

De Falco I (2013) Differential Evolution for automatic rule extraction from medical databases. Appl Soft Comput 13:1265–1283

Del Jesus MJ, Hoffmann F, Navascués LJ, Sánchez L (2004) Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Trans Fuzzy Syst 12:296–308

Derrac J, Garcia S, Sanchez L, Herrera F (2015) KEEL data-mining software tool: sata set repository, integration of algorithms and experimental analysis framework

Gacto MJ, Galende M, Alcala R, Herrera F (2013) Obtaining accurate TSK fuzzy rule-based systems by multi-objective evolutionary learning in high-dimensional regression problems. In: IEEE international conference on fuzzy systems (FUZZ), pp 1–7

Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17:4831–4845

González A, Pérez R (2001) Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Trans Syst Man Cybern Part B Cybern 31:417–425

Hoffmann F (2004) Combining boosting and evolutionary algorithms for learning of fuzzy classification rules. Fuzzy Sets Syst 141:47–58

Hofmann EE, Haskell AE, Klinck JM, Lascara CM (2004) Lagrangian modelling studies of Antarctic krill (Euphausia superba) swarm formation. ICES J Mar Sci J Cons 61:617–631

Ishibuchi H, Nozaki K, Yamamoto N, Tanaka H (1994) Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms. Fuzzy Sets Syst 65:237–253

Ishibuchi H, Murata T, Türkşen I (1997) Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. Fuzzy Sets Syst 89:135–150

Ishibuchi H, Nakashima T, Murata T (1999) Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans Syst Man Cybern Part B Cybern 29:601–618

Ishibuchi H, Nakashima T, Murata T (2001) Three-objective genetics-based machine learning for linguistic rule extraction. Inf Sci 136:109–133

Ishibuchi H, Yamamoto T (2004) Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets Syst 141:59–88

Ishibuchi H, Nojima Y (2007) Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. Int J Approx Reason 44:4–31

Khalili-Damghani K, Sadi-Nezhad S, Lotfi FH, Tavana M (2013) A hybrid fuzzy rule-based multi-criteria framework for sustainable project portfolio selection. Inf Sci 220:442–462

Kromer P, Beshah T, Ejigu D, Snasel V, Platos J, Abraham A (2013) Mining traffic accident features by evolutionary fuzzy rules. In: 2013 IEEE symposium on computational intelligence in vehicles and transportation systems (CIVTS), IEEE, Singapore, 16–19 April 2013, pp 38–43

Liu X, Feng X, Pedrycz W (2013) Extraction of fuzzy rules from fuzzy decision trees: an axiomatic fuzzy sets (AFS) approach. Data Knowl Eng 84:1–25

Liu Y, Qin Z, Shi Z, Chen J (2004) Rule discovery with particle swarm optimization. In: Content computing, ed. Springer, Berlin, pp 291–296

Mansoori EG, Zolghadri MJ, Katebi SD (2008) SGERD: a steady-state genetic algorithm for extracting fuzzy classification rules from data. IEEE Trans Fuzzy Syst 16:1061–1071

Nguyen CH, Pedrycz W, Duong TL, Tran TS (2013) A genetic design of linguistic terms for fuzzy rule based classifiers. Int J Approx Reason 54:1–21

Pedryez W, Vasilakos AV (1999) Linguistic models and linguistic modeling. IEEE Trans Syst Man Cybern Part B Cybern 29:745–757

Price HJ (1989) Swimming behavior of krill in response to algal patches: a mesocosm study. Limnol Oceanograph 34:649–659

Sánchez L, Couso I, Corrales JA (2001) Combining GP operators with SA search to evolve fuzzy rule based classifiers. Inf Sci 136:175–191

Sánchez L, Couso I, Casillas J (2009) Genetic learning of fuzzy rules based on low quality data. Fuzzy Sets Syst 160:2524–2552

Sousa T, Silva A, Neves A (2004) Particle swarm based data mining algorithms for classification tasks. Parallel Comput 30:767–783

Vasilakos A, Ricudis C, Anagnostakis K, Pedryca W, Pitsillides A (1998) Evolutionary-fuzzy prediction for strategic QoS routing in broadband networks. In: The 1998 IEEE international conference on fuzzy systems proceedings, 1998, IEEE world congress on computational intelligence, pp 1488–1493

Wang H, Kwong S, Jin Y, Wei W, Man K-F (2005) Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. Fuzzy Sets Syst 149:149–186

Wilson SW (1995) Classifier fitness based on accuracy. Evol Comput 3:149–175

Yao JT, Vasilakos AV, Pedrycz W (2013) Granular computing: perspectives and challenges. IEEE Trans Cybern 43:1977–1989

Zhang X, Onieva E, Perallos A, Osaba E, Lee V (2014) Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction. Trans Res Part C Emerg Technol 43(Part 1):127–142

Zikidis KC, Vasilakos AV (1996) ASAFES2: a novel, neuro-fuzzy architecture for fuzzy computing, based on functional reasoning. Fuzzy Sets Syst 83:63–84