Designing RBFNs Structure Using Similarity-Based and Kernel-Based Fuzzy C-Means Clustering Algorithms

IRENEUSZ CZARNOWSKI1, (Senior Member, IEEE), JOANNA JĘDRZEJOWICZ2, AND PIOTR JĘDRZEJOWICZ1, (Member, IEEE)

1Department of Information Systems, Gdynia Maritime University, 81-225 Gdynia, Poland
2Department of Informatics, University of Gdańsk, 80-952 Gdańsk, Poland

Corresponding author: Ireneusz Czarnowski (i.czarnowski@umg.edu.pl)

This work was supported by the Gdynia Maritime University.

ABSTRACT The RBF networks belong to a set of artificial neural network architectures. RBF networks have been successfully applied for solving various data mining tasks including classification, and regression. Successful implementation of the RBF network depends on numerous factors among which, the crucial is its structure. The decision on the network structure has to be taken at the network initialization stage. It requires calculating or inducing the number of centroids, and their respective locations. The above problem is known to be NP-hard, and hence, not easily solvable. The traditional approach for deciding on the number of hidden units is based on applying the k-means algorithm for calculating cluster centroids. Unfortunately, the procedure guarantees neither a satisfactory accuracy nor the required generalization level of the RBF network under development. To alleviate the problem for cluster determination, i.e. number of centroids, we propose the similarity-based algorithm (SCA) for the RBF networks initialization, as well as an alternative method for initializing RBFNs using the kernel-based fuzzy clustering algorithm (KFCM-K). In both cases, the number of resulting centroids and their initial locations are provided automatically. The next step involves applying the optimization procedure resulting in the selection of the final centroids’ location. The procedure is integrated with the output weights determination. Since the discussed optimization problem is computationally difficult it has been decided to apply the agent-based population learning algorithm (PLA) which belongs to the class of metaheuristics. A comparative study of approaches based on SCA and KFCM-K is included in the paper. Their effectiveness is demonstrated experimentally using artificial and real benchmark datasets. The results of the computational experiment have shown that both proposed approaches for designing RBFNs perform significantly better than other algorithms used for this task.

INDEX TERMS Radial basis function network (RBFN), RBFN initialization, classification, clustering, multi-agent system.

I. INTRODUCTION

Artificial Neural Networks (ANNs) are computing systems inspired by the biological nervous systems of living organisms. In recent years ANNs play an important role in solving many difficult computational problems. There are a lot of examples, where ANNs have been successfully used to solve difficult problems, including classification, regression, image recognition, time series analysis, etc. Application areas of ANNs have included science, engineering, economy, medicine, and many other fields. The advantage of ANNs over other approximation methods can be attributed to their ability to effectively dealing with a non-linearity in the data [5]. A vast number of neural network architectures have been proposed so far [50]. One such architecture is the Radial Basis Function Network (RBFN). RBFN is an example of the feedforward network which can be seen as a universal approximation tool.

The radial basis function networks usually achieve faster convergence as compared with the multi-layer perceptrons (MLP). Their learning is less computationally complex since a single layer of weights only is required [19].
Convergence speed is not the only factor that makes RBFNs different from MLPs. In comparison to RBFNs, MLP networks have no restrictions on the number of hidden layers and one or more of such layers can be used. Nodes in the hidden and output layers of MLP use identical activation functions, while RBF can use different activation functions at each node. An advantage of the RBF networks is also the fact that, in general, they do not require compiling a large dimensional space of radial basis functions, so the over-fitting effect and as a result, a decrease of their approximation ability is reduced [1]. To sum up, RBFNs are characterized by numerous advantages. In [52] it was underlined that RBFNs have an adaptive structure and generalization capability allowing them to generate high-quality predictions. Additionally, fast convergence to an optimum solution, independence between output values, and primarily allocated weight terms, as well as high accuracy are also features of RBFNs [52].

When we look at the process of RBFN’s computation, the activation functions in the RBF nodes compute a distance between the input examples and nodes centroids, which are often referred to as prototypes. In the case of MLPs, the activation functions compute inner products from the input instances and weights.

A traditional approach to the RBF network design assumes that the network structure is constant in time and that the design process is based on user experience. In fact, in the case of RBFNs, only values of the output layer weights are changing over time. It has been observed that such a traditional approach might not be sufficient enough to deal with non-stationary and complex systems. It is also clear that designing high-quality RBFNs is a complex and difficult task [6], [45], [56].

In [32] it was observed that nodes centroids in an RBFN can be computed using some available clustering algorithms. Such an approach assumes that the internal configuration of the constructed RBFN is directly determined by the clustering algorithm outcome. This could be a disadvantage since centers are selected non-specifically and the RBF performance relies critically on their location. To reduce the risk it was proposed to use the orthogonal least square algorithm, where the RBFN structure can be dynamically changed by adding a new neuron in the hidden layer [32].

Poggio and Girosi in [46] showed that the RBF structure produced by taking centers from all training data, may lead to network overfitting as the number of data becomes too large. To overcome this problem they proposed a network with a limited number of centers, suggesting that it could be useful to apply some clustering algorithms to locate centroids. The cited authors also showed that the updating rule for the RBF centers derived from a gradient descent approach makes them moving towards the majority of data.

During the last 30 years various types of RBFNs, including multiquadric, inverse multiquadric, Duchon radial cubics, thin plate spline, and others have been proposed. Various approaches to designing and modifying RBFNs structure resulting in a more complex structure of hidden layers or the possibility of dynamically changing the hidden layer structure have been also suggested (see, for example, [11], [27], [42]). The problem of the RBF network design is still an active research domain.

The paper focuses on the problem of the RBFNs initialization, however, in the proposed approach, initialization is integrated with the output weights determination. Initialization is the first step of the RBFNs design and has crucial consequences for the quality and performance of RBFNs systems.

Traditionally, in the majority of the reported cases, a well-known k-means clustering algorithm is used to set up the RBF network design parameters, i.e. to produce clusters and set locations of the RBF centroids [7], [18], [40]. This choice is probably inspired by the simplicity of the algorithm and the fact that k-means performs quite well in practice. This simple algorithm copes very well with non-linear features and is quite effective for many problems, including image recognition [47]. Unfortunately, k-means has also several weaknesses. The main disadvantage of the k-means algorithm is its sensitivity to the initial center selection. Poor center location may result in arriving at a poor local minimum [23]. As a remedy, it was proposed to validate at first the obtained clusters, and then optimizing their centers [10].

Another disadvantage of using the k-means algorithm at the initialization stage can be attributed to the fact that the number of clusters in the RBFN design must be set up a priori. In [3] it was shown that there is a strong correlation between clustering decisions and performance of the RBF network.

Several other approaches to the RBFNs initialization have been proposed in [25], [41], [44], [55]. One possible alternative to applying the k-means algorithm is using instead the fuzzy C-means algorithm (FCM) introduced in [49], and extended in [4]. FCM has been successfully used in various applications (see, for example [1], [48]). FCM is based on the association of each cluster with a fuzzy set and with a membership function. This membership function is used for measuring the possibility that data belongs to a given cluster. In comparison to the k-means algorithm, FCM has one important advantage. In the FCM, by fixing the so-called threshold and assuming that all instances having memberships’ degrees exceeding this threshold are assigned to the respective clusters, the number of clusters is determined automatically [37]. Although the FCM has a considerable advantage, in comparison to k-means algorithm, it also has some limitations including its sensitivity to noises [39].

Besides clustering, the support vector machine or the orthogonal forward selection methods have been used for RBFNs initialization. In [29] several strategies for RBFN initialization were reviewed. However, none of the approaches proposed so far can be considered as superior and guaranteeing optimal results in terms of the learning error reduction or increased efficiency of the learning process.

The paper extends earlier research results presented in [15] and [17]. In these papers, the similarity-based clustering algorithm and the kernel-based C-means clustering algorithm...
for the RBF network initialization were, respectively, studied. The current paper focuses on the evaluation and comparison of results using both aforementioned clustering algorithms for the RBF network initialization. Both algorithms provide automatically the number of centroids and their initial locations. In our approach, we do not change the value of this number but locations of centroids are corrected. To solve the resulting optimization problem involving both - determination of centroids locations and network output weights we use the agent-based population learning metaheuristic (PLA) proposed in [2]. The decision to use metaheuristic follows from the fact that the problem at hand is computationally difficult and the agent-based population learning algorithm has been performing well in solving difficult combinatorial optimization problems (see, for example, [13], [14], [27]).

The following sections of the paper include a short overview of the RBF networks design problem, a description of the algorithms used for producing clusters, a presentation of the agent-based population learning algorithm and details of its implementation, a detailed description of the computational experiment setup, and the discussion of the experiment results. The final section contains conclusions and suggestions for future research.

II. RBF NEURAL NETWORKS STRUCTURE AND DESIGN

A. RBF NETWORK STRUCTURE

A radial basis function network is an artificial neural network that uses radial basis functions as activation functions. RBF networks have a three-layer architecture (input, hidden, and output). The input layer consists of a set of units, which connect the network to the external world. The hidden layer consists of hidden neurons where each one uses a radial basis function [19]. An RBF network specificity is that each hidden unit represents a particular point in the data space represented by a set of instances. The output of the hidden unit depends on the distance between the processed instance and a particular point in the input space of instances, represented by a centroid of the basis function. The aforementioned distance is calculated as a value of the activation function, which is the measure of similarity between the considered instance and the respective prototype. The value of this measure is produced through a nonlinear transformation carried-out by the output function, called the radial basis function. For example, the Gaussian function, which is commonly used as the radial basis functions, takes the following form:

$$G(r,p) = e^{-\frac{(r-c)^2}{2}}$$

where $r$ is a norm function denoted as $\| x-c \|$, with $x$ an $n$-dimensional input instance $x \in \mathbb{R}^n$, $c$ represents a centroid and $p$ denotes function parameters.

The output of the RBF network, as a linear combination of outputs of the hidden units, has the following form:

$$f(x) = \sum_{i=1}^{M} w_i G_i(\| x - c_i \|, p_i).$$

where $M$ denotes the number of hidden neurons, $G_i$ denotes a radial basis function associated with $i$-th hidden neuron, $p_i$ is a vector of parameters (like, for example, locations of centroids), $c_i \in \mathbb{R}^n$ are clusters centers (centroids) for $i = 1, \ldots, M$, $\| \cdot \|$ is the Euclidean norm and $w_i \in \mathbb{R}^n$ represents the output layer weight.

B. RBF NETWORK DESIGN

Designing RBFNs involves two stages - initialization and training. The initialization stage requires deciding on the number of hidden units and centroid location. At the training stage connecting weights of outputs of the hidden neurons needs to be estimated.

From the implementation point of view, the initialization stage is considered as crucial from the point of view of the accuracy and stability of the RBF-based approximations [58]. Thus, to design highly accurate and effective RBFN, it is important to apply an effective strategy for deciding on center locations and prototype selection.

Centroids in the RBF network should be selected to minimize the total distance between the data and centroids so that the centers can truly represent the data [37]. A simple way, to measure such distance is to use the square error cost function, which can be defined as:

$$E = \sum_{i=1}^{M} \sum_{j=1}^{N} (\| x_j - c_i \|)^2,$$

where $N$ is the number of instances in the dataset.

During the RBF initialization stage, its centers should be adjusted to minimize the cost function, i.e. the total distance calculated as shown by equation (3). Unfortunately, since the optimization problem in question is computationally difficult, there is a strong risk of getting trapped in a local minimum, especially when using an inappropriate clustering tool. In [38] it has been observed that there is a strong correlation between minimizing the cost function and the overall performance of the RBF network. Lowe in [34] shows that there is no guarantee that the minimum cost function solution will always give the best overall network performance. The above findings stand behind the idea of supporting a clustering algorithm used for designing RBFNs with some constrained optimization algorithms aiming at optimizing the overall RBF network performance. To implement the idea, an algorithm which, could automatically select the RBFN configuration and integrate it with the training process into a single task would be required. Such an algorithm would be particularly useful in the case of using RBFNs for online applications where the network configuration may need changes in time using a self-adjusting structure mechanism [58].

In [16] it was shown that designing the RBF networks through the integration of the initialization and training stages improves, in a majority of cases, network performance. The idea was to carry-out both stages of network development in parallel, using a dedicated middleware framework enabling easy implementation of some optimization algorithm. In the
reported case it was the agent-based population learning algorithm. One of the advantages of the above approach is that the structure of the RBFNs is automatically initialized and locations of centroids within clusters can be if needed, modified during the training process.

To sum up, RBFN design is a complex task. One of the main problems with the RBF neural network is the lack of generally accepted techniques determining how to best implement them [6].

C. COMPLEXITY OF THE RBF NETWORK DESIGN

From the point of view of the computational complexity, the problem of RBFN design belongs to NP-hard class [6], [8], [35], [47].

In [6] it was observed that the overall performance of the RBF network depends on both the output layer weights and the structure of the network. The authors state that the network structure is determined by multiple parameters including the number of nodes, the location of centers, and so-called variances of each node (including the position and shape of each node, respectively). It is also underlined that often the "structure of the network is usually fixed at some compromise settings or based on experience, and only the output layer weights are updated with time". It is also observed that the above approach is not effective especially in nonstationary systems. They also point out that there exist several different approaches where the RBFN structure is constructed dynamically (like in the case of a resource allocating network - RAN, or the extreme learning machine - ELM), but neither of them can guarantee an optimal structure. A drawback in the case of the RAN-like and the ELM-like approaches, as well as in the case of similar ones, is that they do not optimize other RBFNs parameters. As a consequence, node centers tend to be located based on input data characteristics only making the RBFN structure fitting into the data rather than representing a general model of the considered problem. Such a data-centric approach results in increasing the model size as the number of data increases. The final RBF model becomes very large and highly complex. In their discussion on the problem of the RBFN design the authors conclude that it is necessary to optimize the network structure, however, it is, in general, a difficult NP-hard problem [6].

In [8] it was observed that RBFN parameters, including cluster centroids, numbers of nodes as well as output weights, can be trained together via nonlinear optimization. A review of several different approaches for such integrated training is also included in [8]. However, it is underlined that such learning is computationally expensive and may encounter the problem of local minima.

In [47] the complexity of the RBFN design was discussed concerning the setting of the input layer parameters. It was highlighted that the efficiency of the RBF network directly depends on the number of centers. However, the traditional use of k-means for finding centers of the RBF network, which usually performs quite well in practice, has some serious limitations. Their discussion on applying k-means algorithm for RBFNs initialization, the authors conclude that the k-means algorithm is not an optimal one from the computational complexity point of view. The reason is that the RBFN design problem is difficult and belongs to the NP-hard class. In [26] the authors refer to findings in [36], where it was shown that the k-means clustering problem remains NP-complete.

A discussion on the complexity of the RBFNs design can be also found in [28]. The authors comment on learning steps, network structure, and configuration of the network via parameters of the network units, with the conclusion and proof, that the problem of the RBF network design is NP-complete. In particular, the discussed proof shows that it is rather unlikely to expect that any algorithm could be able to find optimal values of weights within a fixed structure of networks in polynomial time. As an example of such a non-polynomial time algorithm, the authors point at the backpropagation procedure.

The complexity of the weights optimization problem in neural networks was also considered in [43]. The authors refer to results of [51] where it was proved that even one neuron network training problem belongs to the NP-hard class.

The problem of the computational complexity of the RBF network design received a lot of attention and has been discussed by many researchers in the past (see, for example, [11], [28], [40], [46]), as well as in more current research (see, for example, [21], [43], [47], [58], [59]). Such a wide interest shows also that RBFNs design is still a lively area of research within the machine learning community and a search for approaches improving process of initialization and training of neural networks, including RBFNs, is still going on.

III. APPROACHES TO CLUSTER INITIALIZATION FOR THE RBF NETWORKS

To assure satisfactory RBF network performance it is needed to optimize its structure. One approach to do this is to identify clusters among instances in the training set. This identification should be carried out independently for each class.

To perform the clustering, we use two approaches. The first is the clustering algorithm based on the similarity coefficient, originally proposed in [12], and the second is a kernel-based fuzzy C-means clustering algorithm. The results of the clustering procedure directly determine the number of network hidden nodes.

A. SIMILARITY-BASED CLUSTERING ALGORITHM

The similarity-based clustering algorithm (SCA) groups instances into clusters. The process is carried out based on the similarity coefficient, calculated for each instance independently. The number of clusters is determined by the value of the similarity coefficient and it is assumed that the similarity coefficient of all instances within a cluster is equal. It means that the number of radial basis functions is set automatically and is equal to the number of produced clusters. To describe the proposed approach formally, the following notation is introduced.
Let \( x \) denote a training instance, \( N \) the number of instances in the training set \( D \) and \( n \) the number of attributes with the total length of each instance (i.e., training example) equal to \( n + 1 \). In the case of training data for classification, the \( n + 1 \) attribute contains the class label that can take any value from the finite set of class labels \( C = \{c_i| i = 1, \ldots, k\} \). Denote the training set as \( D = \{x_{ij}| i = 1, \ldots, N; j = 1, \ldots, n + 1\} \) that is the matrix with \( N \) rows and \( n + 1 \) columns. The pseudo-code of the algorithm producing clusters is shown as Algorithm 1. From the obtained clusters \( C_1, \ldots, C_l \), centroids are used further-on as class and group prototypes.

**Algorithm 1** An Instance Clustering Procedure Based on the Similarity Coefficient Value

**Input:** \( D \) - the matrix containing values of all instances from the training set.

**Output:** clusters from which prototypes can be selected.

1. Transform data instances: each \( x_{ij} \) for \( i = 1, \ldots, n \) and \( j = 1, \ldots, N \) is normalized into interval \([0,1]\) and then rounded to the nearest integer, that is to 0 or 1.
2. Calculate values \( s_i = \sum_{j=1}^{N} x_{ij} \) for \( i = 1, \ldots, n \).
3. For \( l = 1 \) to \( k \) do
4. For \( j = 1 \) to \( N \) do
5. if \( x_{ij,n+1} = 0 \) then
6. \( l_i^l = \sum_{j=1}^{N} x_{ij} s_i \)
7. End if
8. End for
9. End for
10. Map input vectors from \( D \) with the same value of similarity coefficient into clusters
11. Let \( C_1, \ldots, C_l \) denote the obtained clusters satisfying
\[
D = \bigcup_{i=1}^{l} C_i, C_i \cap C_j = \emptyset
\]
12. Return \( C_1, \ldots, C_l \)

From the obtained subsets \( C_1, \ldots, C_l \), prototypes can be selected.

**B. KERNEL-BASED FUZZY CLUSTERING**

Kernel-based fuzzy C-means (KFCM) was introduced to overcome noise and outliers sensitivity in FCM [31]. The idea was based on transforming input data \( D \) into a higher dimensional kernel space via a non-linear mapping \( \Phi \). Such non-linear mapping aimed to increase the possibility of linear scalability of the instances in the kernel space. As a consequence, it allowed performing fuzzy C-means clustering in the feature space.

In the discussed approach, it is assumed that product in the kernel space can be expressed by a Mercer kernel \( K \) as follows [23], [55]:

\[
K(x, y) \equiv \Phi(x)^T \Phi(y).
\] (4)

Such an assumption allows us to carry-out the distance computation in the kernel space using a Mercer kernel function. In this paper, we make use of one of the often-used kernel functions, i.e., the Gaussian function \( K(x, y) \equiv \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right) \) for \( \sigma^2 > 0 \).

Fuzzy clustering is characterized by fuzzy partition matrix \( U = (u_{ij}) \) of size \( t \times N \), where \( u_{ij} \) is the degree of membership of \( x_j \) in cluster \( i \). Standard constraints for fuzzy clustering are assumed, that is:

\[
0 \leq u_{ij} \leq 1, \quad \forall_j \sum_{i=1}^{t} u_{ij} = 1, \quad \forall_i \sum_{j=1}^{N} u_{ij} < N \tag{5}
\]

and the partition of data \( D \) into clusters is carried out using the maximum value of membership degree:

\[
\text{clust}(x_j) = \arg \max_{1 \leq l \leq t} u_{lj}. \tag{6}
\]

In the case of kernel-based fuzzy C-means clustering with prototypes in kernel space (KFCM-K), centroids are approximated by an inverse mapping \( c_i = \Phi^{-1}(v_i) \) from the kernel space to the feature space [24].

The main idea of KFCM-K is to minimize the objective function defined as [9], [22], [57]:

\[
J = \sum_{i=1}^{t} \sum_{j=1}^{N} u_{ij}^m \text{dist}^2(\Phi(x_j), v_i), \tag{7}
\]

where \( m \in \mathbb{N} \) and controls the fuzziness of the memberships. The parameter is fixed and in most cases is equal to 2. From the minimization condition stated as (7) we obtain [57]:

\[
v_i = \frac{\sum_{j=1}^{N} u_{ij}^m \Phi(x_j)}{\sum_{j=1}^{N} u_{ij}^m}, \tag{8}
\]

and the elements in the partition matrix are defined as:

\[
u_{ij} = \frac{1}{\sum_{k=1}^{N} \left(\frac{\text{dist}(\Phi(x_j), v_k)}{\text{dist}(\Phi(x_j), v_k)}\right)^{\frac{m}{m-1}}}, \tag{9}
\]

where:

\[
\text{dist}^2(\Phi(x_j), v_i) = K(x_j, x_j) - 2 \sum_{k=1}^{N} u_{ik}^m K(x_k, x_j) + \sum_{k=1}^{N} u_{ik}^m u_{ij}^m K(x_k, x_j) (\sum_{k=1}^{N} u_{ik}^m)^2 \tag{10}
\]

Finally, to define centroids \( c_i \) the following function is minimized

\[
V = \sum_{i=1}^{t} \text{dist}(\Phi(c_i), v_i), \tag{11}
\]

which in case of Gaussian kernel gives [38]:

\[
c_i = \frac{\sum_{j=1}^{N} u_{ij}^m \cdot K(x_j, c_i) \cdot x_j}{\sum_{j=1}^{N} u_{ij}^m \cdot K(x_j, c_i)}. \tag{12}
\]
Algorithm 2 Kernel-Based Fuzzy C-Means Clustering

**Input:** data \( D \); kernel function \( K \); threshold \( \delta \) for determining the number of clusters.  
**Output:** clusters and their centroids.  
1. For \( l = 1, \ldots, k \) do  
2. Let \( N^l \) be the number of instances from class \( l \), let \( x_1^l, \ldots, x_{N^l}^l \) be the instances from class \( l \).  
3. Let \( K_{ij}^l = K (x_i^l, x_j^l) \) be the quadratic matrix of size \( N^l \times N^l \).  
4. Calculate the eigenvalues of matrix \( (K_{ij}^l) \).  
5. \( r(l) = \text{number of eigenvalues exceeding} \ \delta \).  
6. Initialize \( U^l \) to random fuzzy partition satisfying the conditions (5) and with respect to \( r(l) \).  
7. Repeat  
8. Update \( v_i \) according to (8).  
9. Update \( U^l \) according to (9).  
10. Until stopping criteria are satisfied or the maximum number of iterations reached.  
11. Repeat  
12. Update centroids according to (12).  
13. Until the maximum number of iterations reached.  
14. End for  
15. Map instances from \( D \) according to \( U^l \) \( (l = 1, \ldots, k) \) into \( t \) clusters, where \( t = \sum_{i=1}^k r(l) \).  
16. Let \( C_1, \ldots, C_t \) denote the obtained clusters such that \( D = \bigcup_{i=1}^t C_i \), \( \forall i \neq j, i=1, \ldots, t \ \ C_i \cap C_j = \emptyset \).  
17. Return \( C_1, \ldots, C_t \) and their centroids.

### IV. AGENT-BASED POPULATION LEARNING FRAMEWORK FOR THE RBF NETWORK DESIGN

The RBFN design requires setting values of different parameters related to the network initialization and output weights estimation. Since designing and training of the RBF neural networks can be seen as a combinatorial optimization problem belonging to the NP-hard class [6], [8] using a metaheuristic algorithm seems appropriate.

In this paper, the agent-based population learning algorithm is applied as a collaborative approach to RBF network design, where the process of searching for values of the optimal RBFN parameters is integrated with the process of searching for values of the output weights. The agent-based Population Learning Algorithm (PLA) proposed in [2] belongs to the family of metaheuristic algorithms. Earlier applications have shown that the PLA can produce good quality approximate solutions to a variety of difficult optimization problems [12].

Under the proposed approach clusters are produced at the initialization stage by applying either the SCA or the KFCM-K algorithms. Next, from thus obtained clusters of instances, centroids can be selected, and the agent-based population learning algorithm with a dedicated set of optimization agents is used to locate them. In other words, centroid location is understood as a selection of appropriate instances within each cluster to play the role of centroids.

Thus, the main task of the agent-based population learning algorithm is to determine the values of the following vectors:  
- vector of centroids: \( C = [c_1, \ldots, c_M] \),  
- vector of weights: \( W = [w_1, \ldots, w_M] \).

The proposed approach allows for changing locations of the RBF centers during the training process. It is expected that allowing for dynamic changes of locations could improve the performance of the RBFN-based classifiers since neural network-based classifiers with floating centroids perform better than those with fixed centroids. Computational experiments results shown in [15], [16], and [54] support the above expectation.

### A. AGENT-BASED POPULATION LEARNING ALGORITHM

In [2] it was shown that the population learning metaheuristic implemented within the multi-agent framework can be effectively used in searching for optimal or nearly optimal solutions to complex optimization problems. In the agent-based population learning algorithm implementation proposed in this paper to support the RBF network design, we use the concept of the A-Team originally introduced in [53]. A-Team consists of multiple agents performing various tasks including the most important one, which is improving a solution to the problem at hand. Agents forming an A-Team act asynchronously and cooperate in pursuing the common goal.

An A-Team can be viewed as a set of agents and a set of memories. These agents form a network, however, every agent remains in a closed loop. Agents also possess different skills. There are two main categories of agents. The first as Algorithm 2. For each class independently, first, the number of clusters is calculated and then the clustering is performed.

The KFCM-K produces clusters of prototypes and their centroids. Hence, it can be directly used for the RBF initialization. However, in this paper, it is assumed that centroids can be independently selected from the clusters using techniques described in the next section. The basic parameters and general features of the discussed clustering algorithms are compared in Table 1. The table also contains a comparison of the computational complexity of SCA and KFCM-K.
includes optimization agents. Their skill is a solution improvement or, at least, attempting such an improvement. Optimization agents can use different tools and algorithms. The second category involves agents carrying managerial tasks like maintaining a common memory where current solutions are stored or controlling computation time. A-Team common memory serves as the basis for implicit communication between agents. They read solutions from the common memory and return them to this memory after an attempted improvement. The process of improving solutions is carry-out by optimization agents independently and in parallel.

A-Team implementation of the population learning algorithm executes a sequence of steps shown as Algorithm 3.

### Algorithm 3 A-Team Implementation of the Population Learning Algorithm

1. Generate an initial population of solutions (individuals) and store them in the common memory
2. Activate optimizing agents
3. While (stopping criterion is not met) do (in parallel)
   4. Read individual from common memory
   5. Execute improvement algorithms by optimizing agents
   6. Store individual back in the common memory
4. End while
5. Return Take the best solution from the population as the result

#### B. USING AGENT-BASED PLA FOR DESIGNING THE RBF NETWORKS

A-Team executing the population learning algorithm evolves automatically main components of the RBF network. After completion of the evolutionary process, the emerging network is ready to classify instances with unknown class labels. The learning process requires the training dataset as input. The resulting RBF network, defined by the induced centroids and values of the output weights, is specialized to deal with instances of the same kind as those belonging to the training dataset.

To sum up, the basic features of the proposed approach are as follows:

- Solutions to the RBFN design problem are evolved using the population learning paradigm where individuals in a population are gradually improved by cooperating agents.
- The population of solutions is stored in the common memory shared by all optimization agents.
- A solution is represented by a string (an individual) consisting of two parts. The first contains integers representing indexes of instances selected as cluster centroids. The second contains real numbers representing the output weights.
- The initial population of solutions is generated randomly.
- The fitness of each individual is evaluated by estimating the classification accuracy of the RBFN, assuming it is initialized using the respective centroids and the set of weights described by such individuals.

Individuals are evolved using the following three groups of agents forming the A-Team:

- Optimization agents dealing with the centroid selection. Procedures involve simple local search, random search, and tabu search with predefined sequence modification, replacement, and exchange moves.
- Optimization agents carrying out the procedure for estimation of the output weights implemented as a simple evolutionary algorithm.
- Management agents overlooking communication between optimization agents and the common memory as well as controlling the diversity of individuals stored in the common memory and, eventually, replacing some weak individuals with randomly generated ones.

The main task of optimizing agents is to improve solutions that have been forwarded to them. The solutions are taken from common memory after a random selection. After an improvement attempt, an individual replaces its earlier version or some other worse individual. Replacement is not carried-out if no worse individual can be found in the common memory. Optimizing agents improve the solutions by working independently and asynchronously.

In Fig. 1 the proposed two-step process of the RBFN design is shown.

#### V. COMPUTATIONAL EXPERIMENT

In this section results of the validating computational experiment are shown and discussed.

#### A. COMPUTATIONAL EXPERIMENT SETTING

In the reported experiments the SCA and the KFCM-K have been implemented using the following main settings:

- SCA based clustering with the agent-based population learning algorithm used to select and locate cluster centroids and to estimate network output weights.
- KFCM-K based clustering with the agent-based population learning algorithm used to select and locate cluster centroids and to estimate network output weights.

The above algorithms have been also compared with other implementations:

- K-means clustering with the agent-based population learning algorithm used to locate cluster centroids and to estimate output weights.
- XMeans clustering [26] with the agent-based population learning algorithm used to locate cluster centroids and to estimate network output weights.

The above algorithms have been also implemented with the backpropagation (BP) algorithm for output weights estimation. Thus, two series of experiments have been carried out as shown in Table 4 and Table 5.

The computational experiments have been carried-out using benchmark datasets obtained from the UCI Machine Learning Repository.
Learning Repository [33]. The characteristics of these datasets are shown in Table 2. Each benchmark problem has been solved 50 times, and the experiment plan involved 10 repetitions of the 10-cross-validation scheme. The reported values of the quality measure have been averaged over all runs. The quality measure in the case of the classification problems was the correct classification ratio (accuracy).

The computational experiments and use of the agent-based population learning algorithm have been carried out with parameter settings as shown in Table 3. The values of parameters have been set arbitrarily based on the trial-and-error procedure. In the case of the BF algorithm applied to output weights estimation, the number of epochs during the network training has been set to 1000. The same set of parameters has been also used in our previous researches (see: [15], [17]), results of which are partly used as a reference and for comparison.

In the case of the KFCM-K, the fuzziness of the memberships has been set to 2 and the Gaussian function has been used as a kernel function. Computations within RBFNs hidden units have been also carried out using the Gaussian function, when the dispersion of the radial function was equal to double value of the minimum distance between basic functions [50]. For algorithms based on k-means and random initialization, the number of prototypes has been set in a way assuring comparability concerning the number of prototypes produced by the similarity-based procedure.
To observe the influence of RBFN initialization approaches on the performance of the RBF networks, experiment results have been grouped into 3 categories – small, medium, and large datasets as shown in Table 4.

The JABAT middleware [2], based on JAVA code and JADE (Java Agent Development Framework) [3], has been used as an environment for computational experiments.

### B. EXPERIMENT RESULTS

In Table 4, the mean values of the classification accuracy (in %) obtained using the proposed approach are shown. To compare the proposed approach with alternative ones Table 4 contains also average accuracies obtained using two other clustering algorithms - k-means and the XMeans. The results are also presented for two series of experiments, with PLA and BP applied for output weights estimation. In all cases, the agent-based PLA has been used to locate centroids within clusters, but in cases marked by “PLA”, also to estimate output weights. From the results shown in Table 4, it can be observed that SCA performed best followed closely by KFCM-K. Both algorithms, that is SCA and KFCM-K, outperform in terms of the resulting accuracy of two remaining clustering algorithms. This conclusion seems to hold independently on the dataset size and the algorithm used for output weights estimation.

From the results shown in Table 4, it can be observed that SCA performed best followed closely by KFCM-K. Both algorithms, that is SCA and KFCM-K, outperform in terms of the resulting accuracy of two remaining clustering algorithms. This conclusion seems to hold independently on the dataset size and the algorithm used for output weights estimation. It can be also seen, that PLA outperforms BP for the output weights estimation. To assess more closely the performance of the compared algorithms the Friedman [20] non-parametric test has been carried out. Average computation results obtained using each of the considered clustering algorithms have been ranked and assigned rank value.

The following hypotheses have been formulated:

- $H_0$ – null hypothesis: All of the compared algorithms are statistically equally effective regardless of the kind of problem.
- $H_1$ – alternative hypothesis: Not all algorithms are equally effective.

The analysis has been carried out at the significance level of 0.05. The respective value of the statistics for Friedman’s test with 4 algorithms and 8 instances of the considered problems is 25.083, the value of the distribution is equal to 14.0671 for 7 degrees of freedom. Since the $p$-value of the Friedman test is lower than the considered significance level $\alpha = 0.05$, it can be concluded that there are significant statistical differences between the analyzed results and the null hypothesis should be rejected. From the statistical point of view, it also means that not all considered algorithms are equally effective.

Following the above observation, the post-hoc statistical analysis, based on the Bonferroni-Dunn’s test [20], to detect significant differences between the compared algorithms has been carried out. The value of the critical difference (CD) of Bonferroni-Dunn’s procedure is equal to 3.294563704, computed with the significance level 0.05. The performance of the two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference.

Thus, from the values shown in Table 5, the following can be observed:

- SCA + PLA performs not better than KFCM-K + PLA and SCA + BP,
- KFCM-K + PLA performs not better than SCA + BP and KFCM-K + BP,
- k-means + PLA performs not better than XMeans + PLA, XMeans + BP, KFCM-K + BP and k-means + BP,
- XMeans + PLA performs not better than SCA + BP, k-means + PLA and KFCM-K + BP,
- k-means + BP performs not better than XMeans + BP,
- SCA + BP performs not better than KFCM-K + BP,
- SCA + PLA is close to XMeans + PLA and KFCM-K + BP,
TABLE 5. Differences between the compared algorithms based on the Bonferroni-Dunn’s test.

|       | PLA          | BP           |
|-------|--------------|--------------|
|       | SCA          | KFCM-K       | k-means | XMeans | SCA          | KFCM-K       | k-means | XMeans |
| PLA   | -            | 0.125        | 3.125   | 2.75   | 1.25        | 2.625        | 3.875   | 4.25   |
| KFCM-K| -            | -            | 3       | 2.625  | 1.125       | 2.5          | 3.75    | 4.125  |
| k-means| -            | -            | -       | 0.375  | 1.875       | 0.5          | 0.75    | 1.125  |
| XMeans| -            | -            | -       | -      | 1.5         | 0.125        | 1.125   | 1.5    |
| BP    | SCA          | -            | -       | -      | -           | 1.375        | 2.625   | 3      |
| KFCM-K| -            | -            | -       | -      | -           | -            | 1.25    | 1.625  |
| k-means| -            | -            | -       | -      | -           | -            | -       | 0.375  |

- KFCM-K + PLA is close to k-means + PLA and XMeans + PLA,
- k-means + PLA is close to SCA + BP,
- SCA + BP is close to k-means + BP and XMeans + BP,
- SCA + PLA performs significantly better than k-means + PLA, k-means + BP and XMeans + BP.
- KFCM-K + PLA performs significantly better than k-means + BP and XMeans + BP.

Based on the computational experiment results we can also observe that the SCA and KFCM-K can be used as alternative approaches to the initialization of RBFN and determination of the number of hidden units. The above results confirm also that both alternative approaches assure better accuracy in comparison to k-means algorithm. Although clustering algorithms used in the paper are not optimal in a formal, computational sense, both, that is SCA and KFCM-K are competitive to other earlier proposed approaches. The experiment results confirmed also an interesting finding, that the performance of the RBFN can be improved by adequate determination of centroid locations. In such a case, the selection of appropriate instances to represent the centroids should be considered as the combinatorial optimization problem. Solving such a problem using an appropriate optimization tool can be beneficial to network performance. In our case, the agent-based population learning algorithm turned out to be the right choice. In the reported experiment, RBFNs with output weights calculated using the backpropagation algorithm was outperformed by networks where the PLA has been responsible for output weights settings.

The main reason behind the good performance of the proposed approach is the integration of two main phases of the network RBFN design. Parallel, integrated, and automated centroid selection and weights estimation appear beneficial to the network performance.

VI. CONCLUSION

The paper contributes by proposing an approach to RBF networks initialization and design with learning based on the clustering of the training set of instances. To cluster instances, we propose using the similarity-based clustering algorithm and the kernel-based fuzzy clustering algorithm. Next, at the following stages, cluster centroids and network output weights are selected and estimated by the agent-based population learning algorithm. The computational experiment has shown that both proposed approaches to RBFNs initialization, which are SCA + PLA and KFCM-K + PLA perform well. However, the version with KFCM-K is computationally more complex and requires setting more parameters.

The main advantage of the proposed approach is that the structure and output weights of the RBFN can be automatically generated. The approach can be considered as a step towards the automation of RBFN design.

Future research will focus on studying the influence of additional factors and parameters on the proposed approach to the initialization and design of RBFNs to improve the effectiveness of the approach. It is also planned to improve RBFNs performance by proposing more effective agent-based tools for output weights estimation. Future research direction will also include a more extensive validation of the approach.

REFERENCES

[1] L. E. Aik, T. W. Hong, and A. K. Junoh, “An improved fuzzy C-means clustering algorithm for RBF networks,” Adv. Res. J. Multi-Disciplinary Discoveries, vol. 31, no. 1, pp. 1–5, 2018.
[2] D. Barbucha, I. Czarnowski, P. Jędrzejowicz, E. Ratajczak-Ropel, and I. Wierzbowska, “E-Jabat—An implementation of the Web-based A-team,” in Intelligent Agents in the Evolution of Web and Applications (Studies in Computational Intelligence), vol. 167, N. T. Nguyen and L. C. Jain, Eds. Berlin, Germany: Springer, 2009, doi: 10.1007/978-3-540-88071-4_4.
[3] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, “JADE: A software framework for developing multi-agent applications. Lessons learned,” Inf. Softw. Technol., vol. 50, nos. 1–2, pp. 10–21, Jan. 2008, doi: 10.1016/j.infsof.2007.10.008.
[4] J. C. Bezdek, Pattern Recognition With Fuzzy Objective Function Algorithms. Norwell, MA, USA: Kluwer Academic, 1981.
[5] D. S. Broomhead and D. Lowe, “Radial basis functions, multi-variable functional interpolation and adaptive networks,” Malvern, Worcs, Roy. Signals Radar Establishment, London, U.K., Memorandum 4148, 1998.
[6] H. Chen, Y. Gong, X. Hong, and S. Chen, “A fast adaptive tunable RBF network for nonstationary systems,” IEEE Trans. Cybern., vol. 46, no. 12, pp. 2683–2692, Dec. 2016, doi: 10.1109/TCYB.2015.2484378.
[7] S. Chen, S. A. Billings, and P. M. Grant, “Recursive hybrid algorithm for non-linear system identification using radial basis function networks,” Int. J. Control, vol. 55, no. 5, pp. 1051–1070, May 1992.
I. Czarnowski and P. Jedrzejowicz, “Particle swarm optimization aided orthogonal forward regression for unified data modeling,” IEEE Trans. Evol. Comput., vol. 14, no. 4, pp. 477–499, Aug. 2010, doi: 10.1109/TEVC.2009.2033921.

J. H. Chiang and P. Y. Hao, “A new kernel-based fuzzy clustering approach: Superpixel clustering with cell growing,” IEEE Trans. Fuzzy Syst., vol. 11, no. 4, pp. 518–527, Aug. 2003.

R. Cheruku, D. Edla, and V. Kuppili, “An optimized and efficient radial basis neural network using cluster validity index for diabetes classification,” Int. Arab J. Inf. Technol., vol. 16, no. 5, pp. 816–826, 2019.

R. J. Craddock and K. Warwick, “Multi-layer radial basis function networks: An extension to the radial basis function,” in Proc. IEEE Int. Conf. Neural Netw., Washington, DC, USA, Jun. 1996, pp. 700–705.

I. Czarnowski, P. Jedrzejowicz, and E. Ratajczak, “Population learning algorithm—example implementations and experiments,” in Proc. 4th Metaheuristics Int. Conf., Porto, Portugal, Jul. 2001, pp. 607–612.

I. Czarnowski and P. Jedrzejowicz, “An approach to instance reduction in supervised learning,” in Research and Development in Intelligent Systems XX, F. Coenen, A. Preece, and A. Macintosh, Eds. London, U.K.: Springer, 2004, pp. 267–282.

I. Czarnowski, “Distributed learning with data reduction,” in Transactions on Computational Collective Intelligence IV (Lecture Notes in Computer Science), N. T. Nguyen, Ed. Berlin, Germany: Springer-Verlag, 2011, pp. 3–121.

I. Czarnowski and P. Jedrzejowicz, “Agent-based approach to the design of RBF networks,” Cyber. Syst., vol. 44, nos. 2-3, pp. 155–172, Mar. 2013, doi: 10.1007/s10679-012-9722-v.

I. Czarnowski and P. Jedrzejowicz, “An experimental study of scenarios for the agent-based RBF network design,” in Intelligent Decision Technologies. IDT (Smart Innovation, Systems and Technologies), vol. 39, R. Neves-Silva, L. Jain, and R. Howlett, Eds. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-19857-6_11.

I. Czarnowski and P. Jedrzejowicz, “Kernel-based fuzzy C-means clustering algorithm for rbf network initialization,” in Intelligent Decision Technologies 2016. Smart Innovation, Systems and Technologies, vol. 56, I. Czarnowski, A. Caballero, R. Howlett, and L. Jain, Eds. Cham, Switzerland: Springer, 2016, pp. 337–347.

K.-L. Du, “Clustering: A neural network approach,” Neural Netw., vol. 23, no. 1, pp. 89–107, Jan. 2010.

H. Gao, B. Feng, Y. Hou, and L. Zhu, “Training RBF neural network with hybrid swarm optimization,” in Advances in Neural Networks—ISNN (Lecture Notes in Computer Science), vol. 3971, J. Yi, J. M. Zurada, B. L. Lu, and H. Yin, Eds. Berlin, Germany: Springer, 2006, doi: 10.1007/11755996_86.

S. García, D. Molina, M. Lozano, and F. Herrera, “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: A case study on the CEC’2005 special session on real parameter optimization,” J. Heuristics, vol. 15, no. 6, pp. 617–644, Dec. 2009, doi: 10.1007/s10732-008-9080-4.

S. Gong, W. Gao, and F. Abza, “Brain tumor diagnosis based on artificial neural network and a chaos whale optimization algorithm,” Comput. Intell., vol. 36, no. 1, pp. 259–275, Feb. 2020.

D. Graves and W. Pedrycz, “Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study,” Fuzzy Sets Syst., vol. 161, no. 4, pp. 522–543, Feb. 2010, doi: 10.1016/j.fss.2009.10.021.

N. Grover, “A study of various fuzzy clustering algorithms,” Int. J. Eng. Res., vol. 3, nos. 3, pp. 171–181, Mar. 2014.

T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, “Fuzzy C-means algorithms for very large data,” Neural Comput., vol. 11, no. 4, pp. 1130–1146, Dec. 2012, doi: 10.1162/089976609x412241.

L. Yingwei, N. Sundararajan, and P. Saratchandran, “Performance evaluation of a sequential optimal radial basis function (RBF) neural network learning algorithm,” IEEE Trans. Neural Netw., vol. 9, no. 2, pp. 308–318, Mar. 1998.

M. Mahajan, P. Nimbhorkar, and K. Varadarajan, “The planar k-means problem is NP-hard,” Theor. Comput. Sci., vol. 442, pp. 13–21, Jul. 2012.

M. Y. Mashor, “Adaptive fuzzy C-means clustering algorithm for a radial basis function network,” Int. J. Syst. Sci., vol. 32, no. 1, pp. 53–63, Jan. 2001.

M. Y. Mashor, “Hybrid training algorithm for RBF network,” Int. J. Comput., Internet Manage., vol. 8, no. 2, pp. 50–65, 2000.

A. Mehmoukh, K. Mokrani, and M. Cheriet, “A modified kernelized fuzzy cmeans algorithm for noisy images segmentation: Application to MRI images,” IJCSI Int. J. Comput. Sci., vol. 9, no. 1, pp. 172–176, Sep. 2012.

J. Moody and C. J. Darken, “Fast learning in networks of locally-tuned processing units,” Neural Comput., vol. 1, no. 2, pp. 281–294, Jun. 1989, doi: 10.1162/neco.1989.1.2.281.

A. D. Niros and G. E. Tsekovars, “A novel training algorithm for RBF neural network using a hybrid fuzzy clustering approach,” Fuzzy Sets Syst., vol. 193, pp. 62–84, Apr. 2012, doi: 10.1016/j.fss.2011.08.011.

V. Palade, M. S. Petrov, and T. D. Todorov, “Neural network approach for solving nonlocal boundary value problems,” Neural Comput. Appl., vol. 32, no. 17, pp. 14153–14171, Sep. 2020.

A. Pelaghi, “Global optimization issues in deep network regression: An overview,” J. Global Optim., vol. 73, no. 2, pp. 239–277, Feb. 2019, doi: 10.1007/s10898-018-0701-7.

J. Platt, “A resource-allocating network for function interpolation,” Neural Comput., vol. 3, no. 2, pp. 213–225, Jun. 1991, doi: 10.1162/neco.1991.3.2.213.

M. Pazzouki, S. S. Allaei, M. H. Pazouki, and D. P. F. Moller, “Adaptive learning algorithm for RBF neural networks in kernel spaces,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), pp. 4818–4825, Jun. 2016, doi: 10.1109/IJCNN.2016.777832.

T. Poggio and F. Girosi, “Networks for approximation and learning,” Proc. IEEE, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

Q. Que and M. Belkin, “Back to the future: Radical basis function network revisited,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 8, pp. 1856–1867, Aug. 2020.

A. E. Rasolli, M. Ashtiyani, P. M. Birgani, S. Amiri, P. Mirmohammadi, J. H. Tavakoli, and A. E. Rasolli, “Training a single sigmoidal neuron is hard,” Neural Comput., vol. 42, nos. 1–4, pp. 147–170, 2020.

S. Šíma, “Training a single sigmoidal neuron is hard,” Neural Comput., vol. 42, no. 8, pp. 1091–1097, 2018.

A. Rasolli, M. Ashtiyani, P. M. Birgani, S. Amiri, P. Mirmohammadi, J. H. Tavakoli, and A. E. Rasolli, “Training a single sigmoidal neuron is hard,” Neural Comput., vol. 42, no. 8, pp. 1091–1097, 2018.

E. H. Ruspini, “Numerical methods for fuzzy clustering,” Inf. Sci., vol. 2, no. 3, pp. 319–350, Jul. 1970.
I. Czarnowski et al.: Designing RBFNs Structure Using Similarity-Based and KFCM Clustering Algorithms

[54] L. Wang, B. Yang, Y. Chen, A. Abraham, H. Sun, Z. Chen, and H. Wang, “Improvement of neural network classifier using floating centroids,” Knowl. Inf. Syst., vol. 31, no. 3, pp. 433–454, Jun. 2012.

[55] Y. W. Wong, K. P. Seng, and L.-M. Ang, “Radial basis function neural network with incremental learning for face recognition,” IEEE Trans. Syst., Man, Cybern. B. Cybern., vol. 41, no. 4, pp. 940–949, Aug. 2011.

[56] H. Yonaba, F. Anctil, and V. Fortin, “Comparing sigmoid transfer functions for neural network multistep ahead streamflow forecasting,” J. Hydrologic Eng., vol. 15, no. 4, pp. 275–283, Apr. 2010.

[57] S. Zhou and J. Q. Gan, “Mercer kernel, fuzzy C-means algorithm, and prototypes of clusters,” in Intelligent Data Engineering and Automated Learning—IDEAL (Lecture Notes in Computer Science), vol. 3177, Z. R. Yang, H. Yin, and R. M. Everson, Eds. Berlin, Germany: Springer, 2004, doi: 10.1007/978-3-540-28651-6_90.

[58] Y. Xie, J. Yu, S. Xie, T. Huang, and W. Gui, “On-line prediction of ferrous ion concentration in goethite process based on self-adjusting structure RBF neural network,” Neural Netw., vol. 116, pp. 1–10, Aug. 2019.

[59] Z. Xu, C. Sun, T. Ji, J. H. Manton, and W. Shieh, "Computational complexity comparison of feedforward/radial basis function/recurrent neural network-based equalizer for a 50-Gb/s PAM4 direct-detection optical link," Opt. Exp., vol. 27, no. 25, pp. 36953–36964, 2019.

IRENEUSZ CZARNOWSKI (Senior Member, IEEE) graduated from the Faculty of Electrical Engineering, Gdynia Maritime University. He received the Ph.D. degree in computer science from the Poznan University of Technology and the Ph.D. degree in computer science from the Wroclaw University of Science and Technology. Since 1998, he has been associated with Gdynia Maritime University, where he is currently a Professor in computer science with the Department of Information Systems. His main research interests include the use of artificial intelligence methods in decision support systems and data mining. His research activity is concerned with machine learning, data reduction for machine learning, and data mining. His scientific work also focuses on research in the field of optimization methods and applications of agent systems. He has published more than 100 articles internationally. Since 2016, he has been the Secretary of the Polish branch of IEEE System, Man, and Cybernetics.

JOANNA JĘDRZEJOWICZ graduated in mathematics from Warsaw University. She received the Ph.D. degree from Gdansk University, Poland, and the Habilitation degree in computer science from Poznan University. She is currently a Professor of informatics with the Faculty of Mathematics, Physics, and Informatics, Gdansk University. Her research interests include artificial intelligence, complexity theory, and formal languages and in this area she has published more than 40 articles in the international scientific journals and proceedings, two books, and several textbooks for students. She is also working as a Program Committee Member in a number of international scientific conferences.

PIOTR JĘDRZEJOWICZ (Member, IEEE) received the Ph.D. degree. He is currently a Professor of computer science and the Chair of the Department of Information Systems, Gdynia Maritime University, Poland. His research interests include artificial intelligence and decision support systems. He has published four books and more than 200 articles internationally. He is a member of the Computer Science Committee, Polish Academy of Science. During his career he has been a Visiting Professor in Germany, Great Britain, China, and Sweden, and a Research Fellow with the School of Computer Science, McGill University, Montreal, QC, Canada. From 2012 to 2016, he was served as the Rector of the Gdynia Maritime University and from 2008 to 2012 as the Vice Rector, Research.

* * *