Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1567 – 1583
© TÜBİTAK
doi:10.3906/elk-1909-139

User profiling for TV program recommendation based on hybrid television standards using controlled clustering with genetic algorithms and artificial neural networks

İhsan TOPALLI[1,*], Selçuk KILINÇ[2]
[1]Topalli AI Consultancy Limited, Bognor Regis, West Sussex, United Kingdom
[2]Department of Electrical and Electronics Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, Turkey

Received: 24.09.2019 • **Accepted/Published Online:** 03.01.2020 • **Final Version:** 08.05.2020

Abstract: In this paper, an earlier method proposed by the authors to make smart recommendations utilizing artificial intelligence and the latest technologies developed for the television area is expanded further using controlled clustering with genetic algorithms (CCGA). For this purpose, genetic algorithms (GAs), artificial neural networks (ANNs), and hybrid broadcast broadband television (HbbTV) are combined to get the users' television viewing habits and to create profiles. Then television programs are recommended to the users based on that profiling. The data gathered by the developed HbbTV application for previous studies are reused in this study. These data are employed to cluster users. The number of clusters is found by CCGA, a method proposed in this paper. For each cluster formed by CCGA, a separate ANN is designed to learn the viewing habits of the users of the corresponding cluster. The weight matrices are initialized also by GA. The recommendations produced using the proposed model are then presented by the same HbbTV application developed by the authors. Clustering with GAs gives better results when compared to the well-known K-means clustering algorithm.

Key words: Artificial neural networks, genetic algorithms, hybrid broadcast broadband television, program recommendation, user profiling

1. Introduction

Before the digitalization of TV broadcasts, there was a limited number of TV channels available to viewers and hence viewers had only a few programs shown at the same time to choose among. The developments in technology, both on broadcaster and receiver sides, have enabled larger amounts data to be transmitted, hence increasing the number of channels significantly. Now users have access to more than 100 channels on terrestrial and thousands of channels on satellite. There are at least hundreds of different programs available to users at any given time, which created the modern-age problem for users: "Is there any program that matches my tastes, and if yes, on which channel?"

Monitoring users' viewing habits and recommending television programs based on these habits is not a new topic. However, there is still room for improvement for two main challenges: how to collect data from users and how to use the data to model the users. The most common limitations are data collection for multiple devices and users, and the capacity of the receiver for modeling. Instead of utilizing receivers for excessive calculations, centralizing the intelligence and the logic on the server side would be wiser since cloud computing can handle

*Correspondence: ihsan@topalli-ai.com

big data operations and requests coming from more than one client at a time. The necessity of receiver software modifications is another drawback of current systems, which means having only a particular compatible receiver set to make use of these systems. Evidently, this makes it difficult to expand these recommendation solutions within consumer environments. The IPTV (Internet Protocol Television) platforms where television broadcasts are sent over dedicated IP networks partially address these challenges. IPTV providers usually have dedicated closed network infrastructure where users use a broadband connected set-top box (STB) to receive broadcasts. Each IPTV provider has proprietary requirements for software on the receiver and it is not possible to expand the same recommendation method between providers.

Contrary to proprietary systems, HbbTV brings a standardized method for both ends of the broadcast. The HbbTV framework enables storing choices made by users on a dedicated server controlled by the broadcaster and no additional device is required by the user other than a receiver, which is configured to receive HbbTV applications. According to the HbbTV Association, the number of receivers supporting the HbbTV standard is exceeding 45 million; hence, anyone who has purchased a new TV in the last 8–10 years most likely has a receiver capable of handling HbbTV. Once the application is deployed by broadcasters and the constructed model is running on the server side, all consenting users' data will be collected on the server and these users will be able to receive recommendations via HbbTV, which is already running on their receivers.

In our previous works [1–3], we developed an HbbTV application for collecting users' program-watching history, which was stored and processed on the receiver side. The collected data were employed in the models proposed by the authors [1–3] for user profiling and program recommendation. In this paper, these models are improved further by including a method that we call controlled clustering with genetic algorithms (CCGA). CCGA is a novel clustering technique, which is proposed in this work. The collected data are processed using CCGA to find the optimum number of clusters. The users are grouped into clusters using the CCGA algorithm. Then, for each cluster, a dedicated ANN is constructed and trained in parallel. Instead of taking the common approach of assigning initial weights randomly, these ANNs are started with weights precalculated by a GA. The inputs of the ANNs are the hour of the day, the day of the week, the age group and gender of the user, and program genres, while the output layer is formed by 13 neurons representing the genres and the output is chosen using the softmax function. Once the model has enough data from the users, it will start making recommendations; users will be able to get program information matching their profiles via the same HbbTV application.

The method proposed by the authors is different from others as it utilizes a standardized solution, HbbTV. To the best of the authors' information, utilizing an HbbTV application to gather users' TV viewing patterns has not been done before. Clustering collected data using CCGA is also introduced by the authors. The proposed method is not limited to any brand, manufacturer, or delivery system, which makes it possible to be used with all digital TV receivers. In summary, the main contributions of the paper are incorporating a platform-free approach (HbbTV) for TV program recommendation and using the new CCGA method to cluster collected data, which provide users' TV-watching habits.

This paper is organized as follows: Section 2 provides the theoretical background on the concepts used. Section 3 gives an overview of the related work. The proposal is detailed in Section 4, with subsections on data collection, CCGA, learning, and program recommendation. Performed experiments and obtained results are given in Section 5. Finally, conclusions are drawn in Section 6.

2. Theoretical background

Key concepts and methodologies used throughout this work are HbbTV for data acquisition, ANNs for machine learning, and GAs for clustering and machine learning. The following subsections give brief information on these.

2.1. HbbTV

HbbTV is a worldwide initiative to harmonize the delivery of broadcast and broadband services to users via connected TVs, STBs, and multiscreen devices. Detailed information can be found in the authors' previous work [1] and in the technical specification of European Telecommunications Standards Institute (ETSI) [4].

2.2. Artificial neural networks (ANNs)

The work on ANNs has been very promising in treating nonlinear process modeling. ANNs offer the possibility of studying large complex nonlinear systems. Such nonlinear systems may be based on data that are noisy or dependent on each other. The main operating unit of ANNs is the basic artificial neuron, which can be modeled as a nonlinear device with multiple inputs connected to other neurons by synaptic weights. These synaptic weights are updated according to an adaptive algorithm, which gives the ANN learning capability. By adjusting the synaptic weights, ANNs can produce desired outputs when appropriate inputs are applied [5]. A multilayer perceptron (MLP) is a large class of neural networks with neurons arranged in feedforward layers. In this structure, neurons in adjacent layers are fully connected through unidirectional links called synaptic weights.

2.3. GA for clustering

The GA is a widely used method for data clustering. It can be applied to divide the data into a predefined number of clusters, as well as to detect the number and elements of clusters automatically as proposed by Bandyopadhyay and Maulik [6].

2.3.1. Using predefined number of clusters

In this case it is assumed that the data are to be clustered into a predefined number of sets. The population P consists of N chromosomes, shown as follows:

$$P = \{K_1 \quad K_2 \quad ... \quad K_N\}. \tag{1}$$

The cluster set is:

$$\{C_1 \quad C_2 \quad ... \quad C_M\}, \tag{2}$$

where M is the number of clusters, which is a predefined value. Initial cluster centers are chosen randomly:

$$CC_i = K_j \quad i \in [1, M], j \in [1, N]. \tag{3}$$

Then the cluster center set can be represented as:

$$\{CC_1 \quad CC_2 \quad ... \quad CC_M\}. \tag{4}$$

After determining the centers, chromosomes can be distributed to the clusters based on their distances. For each chromosome, the distance between the chromosome and each cluster center is calculated and the chromosome is assigned to the cluster with the nearest center. These distances between the chromosomes and the cluster centers can be formulated as:

$$D_{ij} = \sum_{x=1}^{L} (K_j(x) - CC_i(x))^2 \quad i \in [1, M], j \in [1, N], \tag{5}$$

where L is the length of the chromosomes. Then, for all Kj, i_{min} is found where

$$D_{i_{min}j} = \min_i D_{ij}. \tag{6}$$

K_j becomes a member of $C_{i_{min}}$:

$$K_j \in C_{i_{min}}. \tag{7}$$

With the chromosomes assigned to the clusters, centers are no longer the real centers of the clusters. They need to be moved to the center of the new members:

$$CC_i(x) = \frac{1}{n_i} \sum_{j=1}^{n_i} K_j(x) \quad i \in [1, M], x \in [1, L], \tag{8}$$

where n_i is the number of chromosomes in the ith cluster. The new cluster center set becomes:

$$\{CC_1 \quad CC_2 \quad ... \quad CC_M\}. \tag{9}$$

At this point, two cluster centers will be selected as parents to produce offspring. However, the parents need to be strong members of the population according to the survival of the fittest phenomenon. Therefore, fitness values of each cluster are to be found. The fitness function is the inverse of the sum of the distances of each cluster member to the center of that cluster, and this fitness function should be maximized. The sum of distances for each cluster is:

$$D_i = \sum_{j=1}^{n_i} \sum_{x=1}^{L} (K_j(x) - CC_i(x))^2 \quad i \in [1, M]. \tag{10}$$

Therefore, the fitness value is:

$$F_i = \frac{1}{D_i} \quad i \in [1, M]. \tag{11}$$

Then the fitness values set becomes the following:

$$\{F_1 \quad F_2 \quad ... \quad F_M\}. \tag{12}$$

Potential parents are the centers of the clusters. Each of them has a chance of selection depending on the fitness value of their clusters. The probability Φ_i of each cluster center against the fitness value F_i can be calculated as roulette-wheel selection:

$$\Phi_i = \frac{F_i}{\sum_{j=1}^{M} F_j} \quad i \in [1, M], \tag{13}$$

where the larger fitness value, the better the chance for selection. Then the set of cluster center probabilities becomes:

$$\{\Phi_1 \quad \Phi_2 \quad ... \quad \Phi_M\}. \tag{14}$$

The parents are chosen within the cluster centers based on their probabilities calculated in the previous step. The cluster center with higher probability is more likely to be chosen as a parent. The first parent is CC_{i_1} with

$$\Phi_{i_1} = CC_{i_1} \quad i_1 \in [1, M], \tag{15}$$

and the second parent is CC_{i_2} with

$$\Phi_{i_2} = CC_{i_2} \quad i_2 \in [1, M] \quad \text{where} \quad i_2 \neq i_1. \tag{16}$$

Offspring production starts with crossing over the parent chromosomes at a random point. After crossover, offspring go through a mutation process with a low probability in order to prevent the algorithm from being stuck in a local minimum. Then the two parents (cluster centers) chosen previously are replaced with the produced offspring. If the cluster centers are modified, then the algorithm continues from forming the clusters. If there is no change, it can be terminated.

2.3.2. Automatic detection of number and elements of clusters

The method given by Bandyopadhyay and Maulik [6] proposes automatic evolution of clusters and their numbers using a GA. In this approach, there is a dataset consisting of the vectors

$$DataSet = \{X_1 \quad X_2 \quad ... \quad X_D\} \tag{17}$$

and a population of chromosomes

$$P = \{K_1 \quad K_2 \quad ... \quad K_N\}. \tag{18}$$

The minimum and the maximum numbers of centers are defined as $K_{\min}$ and $K_{\max}$. Then the population size N is calculated as

$$N = 3(K_{max} - 1). \tag{19}$$

For each chromosome K_j, $j = 1, ..., N$, a random number r_j is assigned. This means the chromosome K_j consists of r_j data points and $(K_{\max} - r_j)$ "don't cares". Then the clusters are formed for each chromosome K_j by assigning data points to r_j clusters corresponding to the closest center. Cluster centroids are found as

$$z_i = \frac{1}{n_i} \sum_{X \in C_i} X \quad i = 1, ...r_j, \tag{20}$$

where n_i is the number of data points in cluster C_i. The scatter within C_i for each chromosome K_j is calculated as

$$S_i = \frac{1}{n_i} \sum_{X \in C_i} \parallel X - z_i \parallel \quad i = 1, ...r_j, \tag{21}$$

and the distance between clusters C_i and C_k is

$$d_{ik} = \parallel z_i - z_k \parallel \quad i, k = 1, ...r_j. \tag{22}$$

The Davies–Bouldin (DB) index for each chromosome K_j is measured to find the fitness value:

$$DB_j = \frac{1}{r_j} \sum_{i=1}^{r_j} \min_{k, k \neq i} \frac{s_i - s_k}{d_{ik}}, \tag{23}$$

$$F_j = \frac{1}{DB_j}. \tag{24}$$

Chromosomes undergo crossover and mutation phases according to the roulette-wheel of DB_j after calculating the fitness values. The old generation is replaced with generated offspring. This procedure is repeated with the new population and K_j with the best fitness is selected. The final number of clusters is equal to the number of data points in K_j. Final members of clusters are assigned around these data points (cluster centers).

3. Literature survey

There are many researchers working on the user profiling problem since the beginning of the 2000s. Program recommendation systems and personalized program guides have been the focus of many studies in recent years. Recommender systems can be classified into three main groups as per Adomavicius and Tuzhilin's overview [7]:

- content-based, where items similar to the ones the users preferred in the past are recommended;
- collaborative, where items that other people with similar tastes and preferences liked in the past are recommended; and
- hybrid approaches, where the above methods are combined.

The hybrid approach of Barragáns-Martínez et al. utilizes singular value decomposition to make television program recommendations in search for a solution to the limitations of content-based and collaborative filtering systems [8]. However, the users need a Web browser to define and enter their preferences and ratings in their system.

The TV Advisor of Das and Horst [9], PTV system of Cotter and Smyth [10], and EPG work of Ardissono et al. [11] are examples of the first television program recommender systems. Among the more recent works, the ones by Wu et al. [12], Tang et al. [13], and Stakhiyevich and Huang [14] can be given.

The TV Advisor of Das and Horst [9] expects TV viewers to specify their preferences to generate recommendations. Cotter and Smyth's PTV uses a mixture of content-based reasoning and collaborative filtering in order to learn user preferences for generating recommendations [10]. Ardissono et al. [11] created the Personalized EPG that employs a module-based system designed for digital receivers. In their work, three user modeling modules collaborate in preparing the final recommendations: Explicit Preferences Expert (preferences declared by the users during initial setup), Stereotypical Expert (users' personal data known to the system and the explicit preferences stated), and Dynamic Expert (users' watching behaviors).

In the construction of recommendation systems, Wu et al. [12] proposed another approach, dual-regularized matrix factorization with deep neural networks, to make use of textual information. They stacked convolutional neural networks and recurrent neural networks and generated representations from their descriptions. Tang et al. [13] proposed a backpropagation neural network recommendation algorithm based on a cloud model. In their algorithm, the user ratings are dealt with by cloud model transformation methods and, at the end, multiple prediction values for users are generated constituting the cloud layer. Using this cloud layer with the neural network, the accuracy of rating prediction is improved. A list of recommendations is generated for the user of concern based on the model. In their work, Stakhiyevich and Huang [14] studied user profiles and their effects on recommendations, in particular for movie recommendation systems. Their method provides recommendations by using similarities between the profile of the user and the features of the movie. Then they make a comparison on how the methods used for creating the user profile affect the provided recommendations.

The papers by Yang et al. [15], Yin et al. [16], and Gao et al. [17] can be given as recent examples on recommendation systems dedicated to the IPTV domain. For a history of the evolution of recommendation engines, the work by Smyth and Cotter can be cited [18].

Another recommendation system example, focusing mainly on movies, is the work of Çataltepe et al. [19], in which different content-based and collaborative recommendation methods for a Turkish movie recommendation system are evaluated and combined. Movie description, actors, directors, years, and genre data are gathered for their recommendation tool. Content features and past implicit ratings acquired from the user are used to produce content feature-based user profiles. Then feature selection is applied on these profiles. Their experimental results show an increase in recommendation success, especially for users who have watched a small number of movies in the past.

The user profiling method proposed in this paper is based on HbbTV. An earlier paper by the authors presented the initial findings of this work [1]. In another paper by the authors, the problem was approached in a similar way but by making use of traditional GA clustering methods only [2]. There is no other user profiling work utilizing HbbTV to the best of the authors' knowledge. Since it is a public standard and it does not require any special software or hardware, it can be realized more widely than the above methods.

Bandyopadhyay and Maulik proposed a method for the evolution of the number of clusters automatically. A new string representation, comprising both real numbers and the "do not care" symbol, is used in order to encode a variable number of clusters [6]. As a measure of the validity of the clusters, the Davies-–Bouldin index is used.

Toz's work [20] proposed an improved form of the ant lion optimization algorithm to solve image clustering problems using a new boundary-decreasing procedure and to obtain well-separated clusters while minimizing the intracluster distances. The experiments showed an improvement in the algorithm and very competitive image clustering results.

In this work, a GA-based algorithm CCGA is proposed to find the number and members of user clusters. It is a modified version of the method mentioned above and it introduces a penalizing transformation for the fitness calculation. This modified version is more appropriate for the time- and size-critical cases as the number of ANNs and the number of learning epochs need to be controlled in the problem presented here.

4. Proposed approach

The proposed approach consists of: 1. data gathering by the HbbTV application (explicit and implicit); 2. automatic detection of number and elements of clusters by CCGA; 3. ANN weight initialization by the GA; 4. ANN training; and 5. program recommendation again by the HbbTV.

The data collection is achieved by the HbbTV application running on the receiver. The HbbTV application sends explicit and implicit data from all the receivers to the cloud. At the server side, the number and the members of clusters are detected automatically by CCGA, a GA-based method proposed in this paper. This is done using the explicitly acquired data. Next, a dedicated ANN is formed for each cluster user.

The weights of these ANNs are initialized using the GA. Then they are trained and tested again on the server by using both types of data. At the end, the same HbbTV application fetches the recommendations from the cloud for each user and shows them on the television screen.

In this model, tasks are shared between the receiver and the server. While the receiver is only responsible for running the HbbTV application for viewing data collection and recommendation presentation, heavy tasks such as learning and preparing recommendations are left to the server. The overall system diagram is given in Figure 1, showing receiver-side and server-side task sharing.

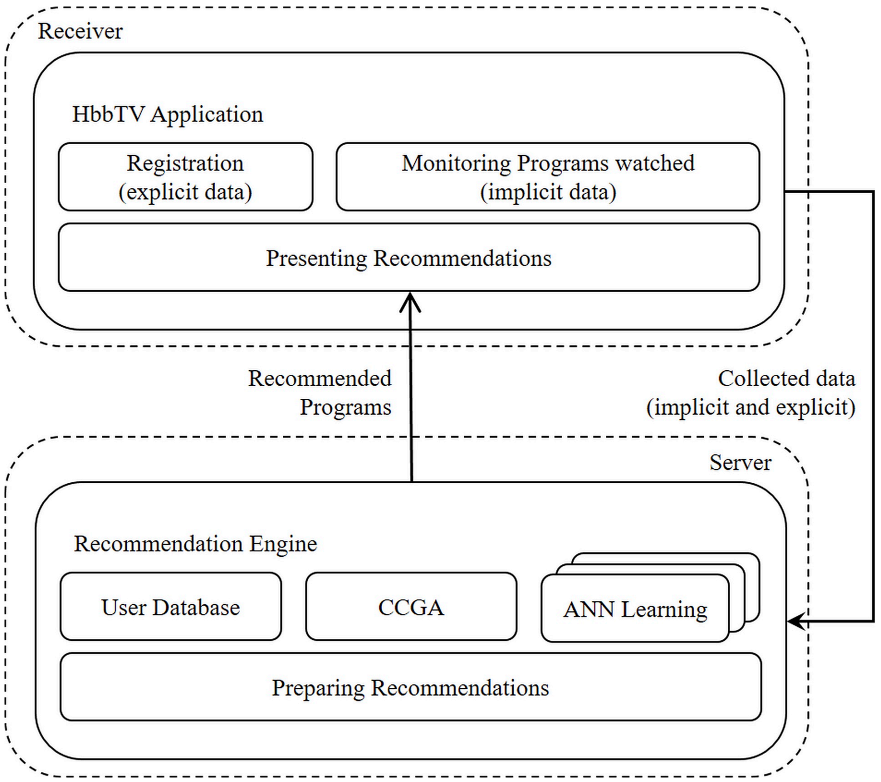
Receiver
HbbTV Application
Registration (explicit data)
Monitoring Programs watched (implicit data)
Presenting Recommendations
Recommended Programs
Collected data (implicit and explicit)
Server
Recommendation Engine
User Database
CCGA
ANN Learning
Preparing Recommendations

Figure 1. Block diagram of the proposed system.

4.1. Phase 1 – Explicit and implicit data gathering

A dataset is mandatory to be able to construct user profiles. As explained in the authors' previous work [1], for all users, the data are to be populated while they are watching any content, such as football match on TV1, news on TV2, or movie on TV3 using the HbbTV application (Figure 2) for all content watched.

Here "explicit data" refers to the information entered by the user directly (age, gender, and preferred genres), whereas implicit data (genres watched) are sent to the server without direct interaction with the user.

4.2. Phase 2 – CCGA: Controlled clustering with genetic algorithms

Since the cluster data are used in the ANN learning in this work, it is important to find the optimum number of ANNs and corresponding I/O sets. Undersized clusters result in large numbers of ANNs, which is inefficient since every new ANN means additional time and cost. Oversized clusters, on the other hand, cause fewer ANNs

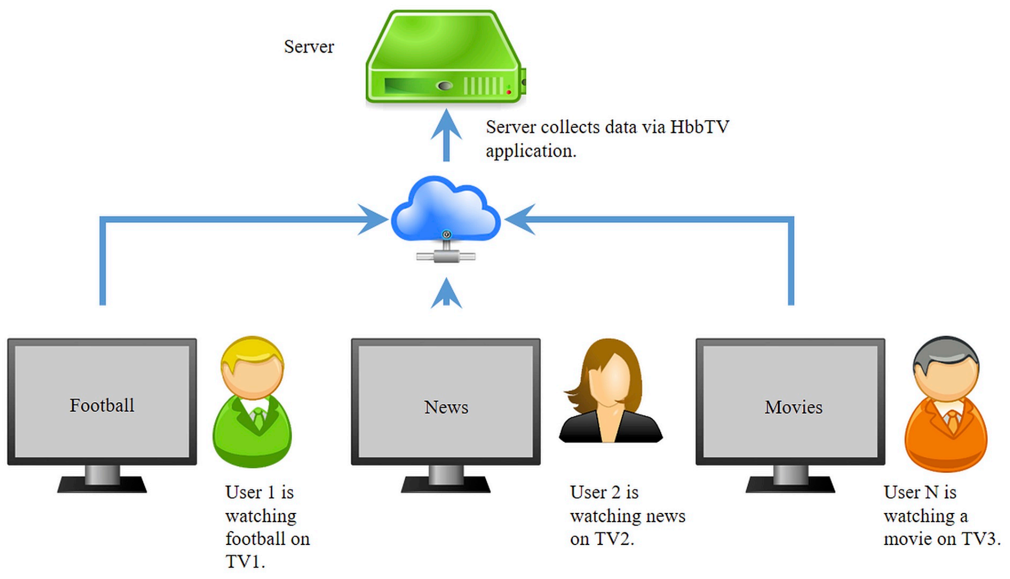
Server
Server collects data via HbbTV application.
Football
User 1 is watching football on TV1.
News
User 2 is watching news on TV2.
Movies
User N is watching a movie on TV3.

Figure 2. Phase 1 - Data collection.

but longer training times, which is again an unwanted situation. Keeping the number of elements in the clusters under control and having a distribution of elements into clusters between the predefined limits is the ideal case. In order to achieve this, a penalizing transformation is introduced for the fitness function calculation.

First the undersized or oversized clusters are identified by setting limit values $n_{\min}$ and $n_{\max}$:

$$n_{\min} \leq n \leq n_{\max}. \tag{25}$$

All the clusters having number of elements n between these minimum and maximum values are considered as ideal cases and no penalty is applied for them. Clusters with number of elements less than the minimum number and clusters with number of elements greater than the maximum number are considered as undersized and oversized, respectively, and undergo a penalizing transformation.

The idea introduced here is to use a penalizing transformation for undersized or oversized clusters to force a smaller fitness function for them. This can be achieved by increasing their DB index, which is equivalent to increasing their scatter value S. Then the new scatter value S' becomes as below. The graphical representation of the penalizing transformation is given in Figure 3.

$$S' = f(S) = \begin{cases} \frac{n_{\min}}{2} & n = 1 \\ \frac{n_{\min}^2 + (2 - n_{\min})n - 2}{2(n_{\min} - 1)} S & n < n_{\min} \\ S & n_{\min} \leq n \leq n_{\max} \\ \frac{n}{n_{\max}} S & n > n_{\max} \end{cases} \tag{26}$$

As seen from the graph, there is no penalty for the clusters whose sizes are between the limits. For outside the limits, a linear penalty is introduced. As cluster size gets further away from these boundaries, the

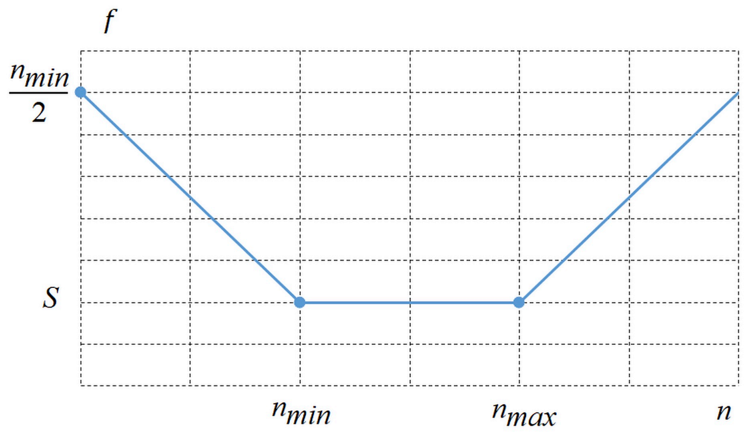

Figure 3. Graphical representation of the penalizing transformation.

induced penalty increases. Then the modified DB index for the kth chromosome becomes

$$DB'_k = \frac{1}{r_k}\sum_{i=1}^{r_k}\max_{j,j\neq i}\frac{S'_i - S'_j}{d_{ij}}, \tag{27}$$

and the modified fitness function is calculated as

$$F'_k = \frac{1}{DB'_k}. \tag{28}$$

The DB index of the clusters whose number of elements is beyond these borders is increased; hence, the fitness function of those clusters is decreased. This means they have less chance of reproduction. Therefore, over the course of GA iterations, the number of elements n in each cluster is limited to a value between a minimum and a maximum figure.

The explicit data (age, gender, and genre preferences) collected in Phase 1 are used as the content of the chromosomes (genes) in CCGA.

Age is represented by a single value, while gender is given as either 0 (female) or 1 (male). Genre preferences are represented by a number between 1 and 13 (Table 1). Then the gene values are normalized between zero and one. For reproduction, single-point crossover is used.

Table 1. Genre values used as genes in the chromosomes.

Genre	Value	Genre	Value	Genre	Value
Turkish movie (TM)	1	Show	6	Social	11
Foreign movie (FM)	2	Sports	7	Education	12
Turkish drama (TD)	3	Children	8	Leisure	13
Foreign drama (FD)	4	Music	9		
News	5	Art	10		

4.3. Phase 3 – ANN weight initialization by GA

In this work, a four-layer MLP model is used. The input layer consists of user information, the time of the day, the day of the hour, gender, age, and available genres of the time of concern. The first and the second hidden layers have 30 neurons each, a value selected close to the number of inputs. The output layer has 13 neurons, representing the number of the genres. The connections between these layers are represented by weights and they need to be initialized before the learning starts.

The general trend is to start learning with random weights. However, it is demonstrated that the weight initialization by the GA gives good results for supervised learning classification problems [21]. Therefore, this approach is taken here. A separate ANN is formulated for each cluster and the weights of each ANN are initialized by the GA.

Parents are selected according to roulette-wheel algorithm. After single-point crossover and mutation with 1% probability, offspring are created and replaced by the parents. The resulting weights are stored to be used in the next phase as the ANN initial weights.

4.4. Phase 4 – ANN learning

After the weight initialization, ANN learning takes place. Constructed ANNs are trained and tested several times with randomly selected 90% and 10% of the data collected by the HbbTV application, respectively, to avoid overfitting.

During each epoch, randomly ordered training data are fed into the neural networks and after each input/output data element, the weights are updated. At the end of each epoch, test data are fed into the network and error is calculated without changing the weights. The average test error is compared against the previously calculated minimum test error, and if lower, this epoch's average test error becomes the new minimum test error. Hence, this set of weights is stored. Learning is terminated either at a predetermined epoch or at a test error below a predetermined value and stored weights are used for the program recommendation phase.

It is known that the learning of neural networks is effective if the learning rate, η, is appropriately chosen. A too small η makes learning impractically slow and is therefore not useful and a too large η spoils the convergence of learning. Therefore, in this work the learning rate is taken as 0.5. The slope of the sigmoid, γ, defines the shape of the sigmoid. Larger values make the sigmoid closer to a step function. Hence, it is taken as 0.9 in this study. The momentum factor, α, is used to smooth the weight changes in order to improve learning. As α gets closer to 1, more weight is given to the past gradients. Here, it is used as 0.9. All the values used for learning rate, η, sigmoid slope, γ, and momentum factor, α, are typical values. They are chosen as "balanced" figures, which cause neither overshoot nor slow convergence. Learning starts with the weights initialized by the GA in the previous phase.

In this model, ANN learning immediately starts when at least one user has watched at least ten programs from a genre. Other users and genres are gradually included in the training and testing sets when they have reached at least ten data. Learning is performed on a daily basis (e.g., every night at 03:00 am) so that the new users of that day can be included in the system. ANN weights of minimum test errors are saved for the next day's training.

4.5. Phase 5 – Program recommendation

The ANN model successfully constructed after the learning phase is ready to make recommendations suitable to the user's viewing tastes. Suggestions are provided to the user via the same HbbTV application, which communicates with the "Recommendation Engine" in the cloud (Figure 4). The recommendation engine uses the trained and tested ANN of the cluster that the user belongs to and calculates the output vector by providing the necessary inputs and using the stored weights. The greatest value of the output vector is considered as the ANN output and the genre associated with that output neuron is taken as the genre to be recommended. Then the recommendation engine lists the current programs for which the genre matches with the ANN output, and the HbbTV application fetches this list to present to the user.

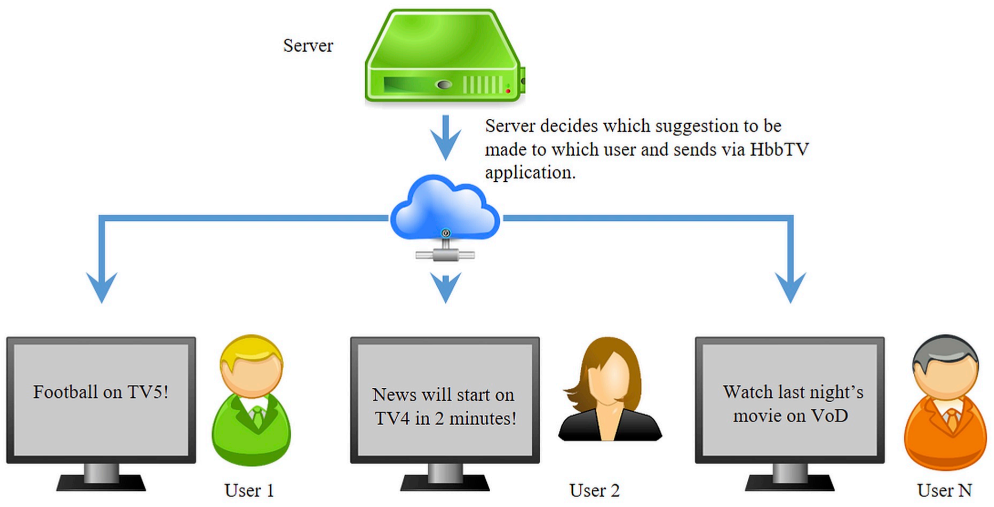
Server
Server decides which suggestion to be made to which user and sends via HbbTV application.
Football on TV5!
User 1
News will start on TV4 in 2 minutes!
User 2
Watch last night's movie on VoD
User N

Figure 4. Phase 5 - Program recommendation.

5. Experiments and results

Utilizing an HbbTV application to collect data has practical difficulties such as adding that HbbTV application to the real broadcast signal. Therefore, the data are collected using another approach as if they came from the HbbTV application. It is done by an online questionnaire where participants provide age, gender, and genre preferences. These data are used as the explicit information. As the implicit information, data are generated from the questionnaire responses. The logic for data generation is the same as in the authors' previous work [3].

There were 248 people participating in the questionnaire. Genderwise, female participants (101 people, 41%) were slightly less than male participants (147 people, 59%). With these data, three experiments are conducted:

- Experiment I: Clustering 248 people to a predefined number (eight) of sets by GA, and then training eight ANNs with these clustered sets with random initial weights.
- Experiment II: Finding the numbers and members of clusters automatically and clustering 248 people to that number (17) of sets by CCGA, and then training 17 ANNs with these clustered sets with GA-based initial weights.
- Experiment III: Clustering 248 people to a predefined number (eight) of sets by K-means algorithm and then training eight ANNs with these clustered sets.

5.1. Experiment I

In this experiment, the users are divided into clusters by GA before the ANN learning. The people who responded to the questionnaire form the population. Therefore, the size of the population is 248. The number of clusters is determined as eight. Within 200 iterations the GA has converged and formed the final clusters. Then each cluster is trained with a separate ANN. The recommendation results for each of them are presented in Table 2.

Table 2. Recommendation results for Experiment I.

Cluster	Number of people	Recommendation error (%)
1	18	0.23
2	51	0.39
3	34	0.55
4	46	0.98
5	23	0.87
6	34	0.92
7	27	1.17
8	15	1.37
Weighted avg.		0.77

5.2. Experiment II

In the previous section, the number of clusters is fixed and chosen as eight. Here, it is demonstrated to find the number and the elements of the clusters automatically by CCGA. The dataset consists of 248 people who participated in the questionnaire:

$$DataSet = \{X_1, X_2, , X_{248}\}. \tag{29}$$

The minimum and maximum numbers of centers are chosen as:

$$K_{\min} = 2, \tag{30}$$

$$K_{\max} = 100. \tag{31}$$

Then the population size N is calculated as

$$N = 3(K_{\max}1) = 297. \tag{32}$$

The penalizing transformation limits $n_{\min}$ and $n_{\max}$ are taken as 10 and 30, respectively. The chromosomes include age, gender, and five genre preferences three different times: weekday evenings, weekend daytime, and weekend evenings. According to this algorithm, the final number of clusters is 17; hence, the population is divided into 17 clusters. Table 3 shows the distribution.

As expected, all cluster sizes are between the minimum and maximum limit values, $n_{\min}$ and $n_{\max}$. This shows that CCGA has reached its aim with the penalizing transformation introduced. Then 17 ANNs are formed and trained with the weight matrices initialized using the GA. Table 4 gives the recommendation results obtained after 23 ANN learning epochs.

Table 3. Clustering results with CCGA.

Cluster	Number of people	Cluster	Number of people
1	15	10	11
2	14	11	15
3	22	12	15
4	17	13	16
5	11	14	13
6	10	15	11
7	15	16	17
8	17	17	17
9	10	Total	248

Table 4. Recommendation results for Experiment II.

Cluster	Recommendation error (%)	Cluster	Recommendation error (%)
1	1.64	10	0.92
2	1.11	11	0.58
3	0.57	12	0.42
4	0.69	13	0.44
5	0.44	14	0.54
6	0.69	15	0.97
7	1.01	16	0.63
8	0.39	17	0.95
9	0.46	Weighted avg.	0.72

5.3. Experiment III

K-means is one of the simplest and widely used unsupervised learning algorithms. Therefore, it is chosen as the comparison algorithm with the GA. The difference between GA and K-means clustering is that there are no fitness calculation, crossover, or mutation steps in K-means. The results with K-means clustering are shown in Table 5.

By comparing the results of the three experiments, one can say that data clustering by CCGA and ANN weight initialization by GA is a better approach than K-means clustering, predefined numbers of clusters, and random initial weights [3].

6. Conclusion and future work

In this paper, an improved system to collect program genres that users have watched and to make recommendations in accordance with their viewing history by utilizing HbbTV is proposed. A complete method, including data collection at the client side, intelligent learning at the server side and program recommendation again at the client side is proposed, implemented, and validated. This method is shown by developing an HbbTV application.

Here, the link between the data acquired and programs recommended is constructed with the GA and ANN. For the proposed method to work well with increasing numbers of users, the ANN learning is improved

Table 5. Recommendation results for Experiment III.

Cluster	Number of people	Recommendation error (%)
1	40	1.00
2	37	1.63
3	29	1.03
4	27	0.68
5	23	2.39
6	28	1.04
7	41	1.21
8	23	1.40
Weighted avg.		1.27

by clustering the users. The data of the users with similar features are clustered into smaller subsets and a separate ANN is created for each cluster. The number and the members of clusters are found by CCGA, a method based on GA, which improves the previous algorithm for cost- and time-critical processes.

This proposed CCGA-based clustering system is tested with the data collected from 248 people. The CCGA method determines the optimum number of clusters as 17 and these clusters are formed accordingly. Then 17 ANNs are constructed to be trained and tested with the respective cluster data. The initial weights of these ANNs are calculated also by the GA. The recommendation results with CCGA show that finding the number and the elements of clusters automatically gives smaller errors than using the predefined number of clusters.

One should note that the dataset collected by the online questionnaire would be significantly different than the dataset collected by the real HbbTV application. Necessary modifications should be made to the algorithm proposed here in case a real-time HbbTV dataset is gathered.

The proposed method is tested based on the assumption that for each receiver there would be a single user. However, in a real home environment, it is most likely that there would be more than one viewer using the same receiver. In multiple-user environments, the HbbTV application can be modified to present a user selection screen when the receiver is turned on. Then the database on the server would have different tables for each user even if the data come from a single receiver.

It is also possible to extend the recommendations given to the users by the HbbTV application. It can present recommendations not only based on that particular user, but also the most watched program at that time in the same region, by the people in the same age group, same gender, etc.

It is also possible to get more detailed information regarding the genre (not just “Sports” but “Football” as well) from the broadcast and the extracted data can be used to improve the recommendations. Once the data are received by the server, it is also possible to enrich the data with more details. As the server is controlled by the broadcaster, information regarding the teams involved in a football match or director or actor information for a movie can be added to the database easily, improving the recommendation.

HbbTV can also be used for other purposes than providing recommendations to the user. Broadcasters will have an enormous amount of data on their servers, which can easily be used to find out which content has been watched most or which channel has been the users’ favorite. The data can be used for providing better user experiences as well as getting more revenue for commercials.

References

[1] Topalli I, Kilinc S. Modelling user habits and providing recommendations based on the hybrid broadcast broadband television using neural networks. IEEE Transactions on Consumer Electronics 2016; 62 (2): 182-190.

[2] Topallı İ, Kılınç S. A smart program recommender system based on the hybrid broadcast broadband television. Dokuz Eylül University Faculty of Engineering Journal of Science and Engineering 2018; 20 (58): 64-74.

[3] Topallı İ. Modelling user habits and providing recommendations based on hybrid television standards using artificial neural networks together with genetic algorithms. PhD, Dokuz Eylül University, İzmir, Turkey, 2017.

[4] ETSI. ETSI TS 102 796 v1.3.1: Hybrid broadcast broadband TV. Paris, France: ETSI, 2015.

[5] Haykin S. Neural Networks: A Comprehensive Foundation. Upper Saddle River, NJ, USA: Prentice Hall International Inc., 1999.

[6] Bandyopadhyay S, Maulik U. Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition 2002; 35 (6): 1197-1208.

[7] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 2005; 17 (6): 734-749.

[8] Barragáns-Martínez AB, Costa-Montenegro E, Burguillo JC, Rey-López M, Mikic-Fonte FA et al. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Information Sciences 2010; 180 (22): 4290-4311.

[9] Das D, Horst H. Recommender systems for TV. In: Proceedings of 15th AAAI Conference; Madison, WI, USA; 1998. pp. 35-36.

[10] Cotter P, Smyth B. PTV: Intelligent personalised TV guides. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence; Austin, TX, USA; 2000. pp. 957-964.

[11] Ardissono L, Gena C, Torasso P, Bellifemine F, Difino A et al. User modeling and recommendation techniques for personalized electronic program guides. In: Ardissono L, Kobsa A, Maybury MT (editors). Personalized Digital Television. Dordrecht, the Netherlands: Springer, 2004. pp. 3-26.

[12] Wu H, Zhang Z, Yue K, Zhang B, He J et al. Dual-regularized matrix factorization with deep neural networks for recommender systems. Knowledge-Based Systems 2018; 145: 46-58.

[13] Tang H, Lei M, Gong G, Wang J. A BP neural network recommendation algorithm based on cloud model. IEEE Access 2019; 7: 35898-35907. doi: 10.1109/ACCESS.2018.2890553

[14] Stakhiyevich P, Huang Z. An experimental study of building user profiles for movie recommender system. In: Proceedings of the 21st IEEE International Conference on High Performance Computing and Communications; China; 2019. pp. 2559-2565. doi: 10.1109/HPCC/SmartCity/DSS.2019.00358

[15] Yang C, Ren S, Liu Y, Cao H, Yuan Q et al. Personalized channel recommendation deep learning from a switch sequence. IEEE Access 2018; 6: 50824-50838. doi: 10.1109/ACCESS.2018.2869470

[16] Yin X, Chen Y, Mi X, Wang H, Tang Z et al. Time context-aware IPTV program recommendation based on tensor learning. In: 2018 IEEE Global Communications Conference; Abu Dhabi, United Arab Emirates; 2018. pp. 1-15. doi: 10.1109/GLOCOM.2018.8647211

[17] Gao Y, Wei X, Zhang G, Zhou L, Dong Z. Mining IPTV user behaviors with an enhanced LDA model. In: 2018 IEEE Global Communications Conference; Abu Dhabi, United Arab Emirates; 2018. pp. 1-20. doi: 10.1109/GLOCOM.2018.8647754

[18] Smyth B, Cotter P. Case-studies on the evolution of the personalized electronic program guide. In: Ardissono L, Kobsa A, Maybury MT (editors). Personalized Digital Television. Dordrecht, the Netherlands: Springer, 2004. pp. 53-71.

[19] Çataltepe Z, Uluyağmur M, Tayfur E. Feature selection for movie recommendation, Turkish Journal of Electric Engineering & Computer Science 2016; 24: 833-848. doi: 10.3906/elk-1303-189

[20] Toz M. An improved form of the ant lion optimization algorithm for image clustering problems. Turkish Journal of Electric Engineering & Computer Science 2019; 27: 1445-1460. doi: 10.3906/elk-1703-240

[21] Maulik U, Bandyopadhyay S. Genetic algorithm-based clustering. Pattern Recognition 2000; 33: 1455-1465.