MODEL-BASED TEST CASE PRIORITIZATION USING NEURAL NETWORK CLASSIFICATION

Nida Gökçe[1], Mübariz Eminli[2]

[1]Department of Statistics, Muğla Sıtkı Koçman University, Muğla, TURKEY
[2]Department of Computer Engineering, Halic University,
İstanbul, TURKEY

ABSTRACT

Model-based testing for real-life software systems often require a large number of tests, all of which cannot exhaustively be run due to time and cost constraints. Thus, it is necessary to prioritize the test cases in accordance with their importance the tester perceives. In this paper, this problem is solved by improving our given previous study, namely, applying classification approach to the results of our previous study functional relationship between the test case prioritization group membership and the two attributes: important index and frequency for all events belonging to given group are established. A for classification purpose, neural network (NN) that is the most advances is preferred and a data set obtained from our study for all test cases is classified using multilayer perceptron (MLP) NN. The classification results for commercial test prioritization application show the high classification accuracies about 96% and the acceptable test prioritization performances are achieved.

KEYWORDS

Test case prioritization, event-based modeling, neural network classification, model-based prioritization

1. INTRODUCTION

Software testing involves identifying the test cases which find the faults in the software. It is the most important techniques for achieving high quality software systems [1]. However, it is very time consuming and expensive process. Therefore, many researchers have proposed diverse techniques to test the entire software with limited time and resources. These techniques are called as test suite minimization, test suite reduction, test case selection and test case prioritization [2]. Test suite minimization and test suite reduction techniques remove redundant test cases to decrease the size of the test suite. Test case selection techniques select some of the test cases which are the changed parts of the software. Test case prioritization (TCP) techniques rank the entire test cases in the test suite to maximize some properties, such as the rate of fault detection or coverage rate [3].

The existing TCP techniques are generally proposed in regression testing which is to re-run all existing test cases in previous version of the software. While both the test suite minimization and test case selection techniques decrease testing time and cost, they can exclude some critical test cases that can detect the faults [4]. However, TCP techniques use the entire test suite and decrease testing cost by achieving parallelization of the debugging and testing activities [5]. They can be roughly classified into two: code-based and model based. The code-based TCP uses source code of the software system to prioritize test cases. The model-based TCP uses the model that represents the desired behavior of software system to prioritize test cases. System modeling can

DOI : 10.5121/cseij.2014.4102

use to model state-based or event-based systems, e.g. embedded systems [6]. There are several modeling languages in literature: Extended Finite State Machine, Specification Description Language, UML (Unified Modeling Language) sequences or activity diagrams, state charts and ESG (Event Sequence Graphs) [4, 6, 7]. ESGs differ from the other modeling techniques, represent the behavior of a system in interacting with the user's actions. In an ESG, events are ones which are in accordance with the user expectations. The ESG model is used to generate the test scenario called as test suite [7].

The approach focused on this article, event-based modeling, is also different from black-box and other model based testing approaches. They use TCP models that are based on "usage models" and "usage profiles", which might not be available for all existing software.

In our previous study, ESG model based TCP with using clustering approach has been presented in order to reduce above mentioned constraints [8,9]. This approach aims to determining indirectly preference degree of test cases which are usually given by several attributes of relevant events through clustering all events entailed and frequency of occurrence of events, and then ranking them in according to preference degrees obtained. It is obvious that such as a way test prioritization has high computational cost for a large number of test cases in the one hand and it is capable to ranking the existing test cases only on the other hand. Whereas there may be needed updating or extending test suite by adding new ones afterwards

In this paper improving our previous study [8-10] new model-event based TCP is suggested where instead of ranking of test cases in according to their preference degree, they are divided into preference groups through the use classification technique. For classification, each test case is presented as a data point where previously obtained important index and frequency of occurrence for all events belonging to given test case as attributes and its corresponding preference label as class membership are represented. A data set formed in such a way for all test cases are classified by using multilayer perceptron (MLP) neural network (NN).

In this study, we investigate whether the use of neural network classification approach can help improve the effectiveness of the model-based TCP technique, during the modified part of the system testing especially. We perform a new TCP technique combined a neural network classification approach. Proposed approach aims to determine the critical test cases that can early detect the faults of the software, without needs any extra test cost. By way of neural network training, the suggested approach an assign any test case to the priority group to be defined in advance.

The rest of the paper is organized as follows. Section 2 discusses background information and related work relevant to TCP techniques. Section 3 describes our new test case prioritization technique that incorporates a neural network classification approach. Section 4 presents our experiments, results and analysis. Finally, Section 5 presents conclusions and discusses possible future work.

2. BACKGROUND AND RELATED WORKS

In this section, TCP problem is briefly explained and the methodology is introduced.

Software testing is a time-consuming and expensive process [1]. The goal of testing is to validate changes in a version of software during software maintenance. Software developers need to make sure that modifications are correct and do not adversely affects the modified version of the

software. It is therefore important to decide which part of the software should be tested first. For this purpose, test case selection, test suite reduction, and test case prioritization techniques are suggested in literature [4, 11-13].Test suite minimization and test suite reduction techniques remove redundant test cases to decrease the size of the test suite. Test case selection techniques select some of the test cases which are the changed parts of the software. Test case prioritization (TCP) techniques rank the entire test cases in the test suite to maximize some properties, such as the rate of fault detection or coverage rate [3].

TCP problem is firstly defined by Rothermel et al. [3] as follows:

Given: A test suite T; T′and T″ are different variation of the test suite; the set PT of permutations of T; a function f from PT to the real numbers, which represents the preference of the tester while testing.

Problem:

$$\text{Find} T' \in PT \text{such that} (\forall T'')(T'' \neq T')[f(T') \geq f(T'')]$$

Rothermel et al. studied several prioritizing techniques which based on coverage of various structural elements of software such as total statement, additional statement coverage, branch, additional branch coverage and fault-exposing potential. They showed that the prioritization techniques can improve the rate of fault detection [3]. Wong et al. introduced a TCP technique with the main focus on the criterion of increasing cost per additional coverage [14]. Kim and Porter prioritized test cases based on historical execution data [15]. They handled an experiment to assess its effects on the long run performance of resource-constrained regression testing. Srikanth et al. proposed a value-driven method to prioritize the system level test cases, called the Prioritization of Requirements for Test (PORT) [16]. This technique was based on requirements volatility, customer priority, implementation complexity and fault proneness of the requirements. Krishnamorthi et al. prioritized both new and regression test cases [17]. Srivastava and Thiagarajan studied a prioritization technique that was based on the changes that have been made to the program and focused on the objective function of impacted block coverage [18]. Sabharwal et al. proposed a technique to prioritize test case scenarios by identifying the critical path clusters using genetic algorithm [19]. They derived test cases from the UML activity diagram and state chart diagram. Korel et al. proposed a model-based test reduction technique that uses Extended Finite State Machine (EFSM) model dependence analysis to decrease a given regression test suite. They also performed an experimental study in order to compare simple code-based and model-based TCP techniques [6]. The results have shown that model-based test prioritization may significantly improve the early fault detection as compared to the code based test prioritization. This method however needs prior information of the system (e.g., source code, number of faulty etc.) and it could only be used in system retesting. The model- based techniques of TCP use only the behavioral model of given system the purpose of testing. To obtain such a model, several modeling language have been developed, including EFSM, Specification Description Language (SDL), event flow graphs, UML diagrams and Event Sequence Graphs (ESG) [4,10-13].

There are many TCP techniques in the literature. Yoo and Harman (2012) published a survey on regression test minimization, selection and prioritization [2]. They investigated 47 TCP papers published 1999 and 2009 and classified the existing research on prioritization into the following groups: code coverage-based prioritization, distribution-based prioritization, human-based prioritization, requirement-based prioritization, history-based prioritization, model-based prioritization, fault-exposing-potential prioritization etc. Catal and Mishra (2013) published a systematic mapping study on test case prioritization [4]. Their work is derived from the analysis of a systematic search of journal papers and conference proceedings, which led to identification

of 120 TCP papers. They showed that coverage–based TCP techniques dominate (40%). In addition, greedy-based and model based techniques are generally preferred. The proportion of papers on model-based techniques is increasing (12%), but at slow rate [4]. Some researchers have demonstrated that model-based TCP techniques can outperform code-based TCP techniques [6].

In our previous works, we suggested model-based TCP techniques using cluster analysis which are both neural network clustering and fuzzy clustering [8, 9, 20]. These techniques are discussed and formally described in [8, 9, 20].They used ESGs for the purpose of modeling, under the assumption that the behavior of system has correctly been specified and modeled by a set of ESGs. Test scenario derived from ESG models. To generate test cases from ESGs, arc coverage is used as the criterion. Arc coverage is essentially a measure of to what degree all possible sequences of events have been covered. For clustering purpose, each event is treated as if it is a point in multidimensional space, one dimension for each attributes. For forming importance groups of events, events are clustered using unsupervised NN and fuzzy c-means clustering algorithm. After obtained clusters of events, an importance degree is assigned to every event in each cluster using the importance degree of the cluster. The preference degrees are defined by taking into account the importance degree of events. Finally, the priorities of test cases are determined by ranking the calculated preference values from highest to lowest [9-10]. One of the advantages of these techniques needs no prior knowledge, such as number of faults, usage profiles, binary or source code of the system under test, which in fact makes the method radically different from the most existing approaches.

In this paper, as extension to our previous works we propose a technique for prioritizing test cases from ESG using neural network classification. The proposed approach focused on performances in the training of feed-forward back-propagation NN was proposed by Rumelhart at al. [21].

3. TEST CASE PRIORITIZATION WITH NN CLASSIFICATION

This section illustrates the details of our proposed approach for TCP using feed-forward back-propagation NN.

3.1. NN Classification

Classification is the most researched topic of neural networks. In neural network, a neuron is the basic component and a multi-layer perceptron structure is a fully interconnected set of layers of neurons. Each neuron of a layer is connected to each neuron of the next layer, from the input layer to the output layer through the hidden layers.

Feed-forward back-propagation (FFBP) learning algorithm is famous learning algorithm among NNs for classification. In the learning process, to reduce the inaccuracy of NNs, FFBP use the gradient-decent search method to adjust the connections.

Multi-layer perceptrons (MLP) are a popular form of feed-forward neural networks with many successful applications in data classification [22]. The supervised learning (training) process of an MLP with input data and target, requires the use of an objective function (or error, cost, loss function) in order to assess the deviation of the predicted output values, from the observed data values t and use the assessment for the convergence towards an optimal set of weights [22].Feed-forward networks generally use Levenberg-Marquardt algorithm, because it is the fastest training algorithm. An iterative algorithm does the adjustment during the training phase of the NN. During training phase, a selected set of training data is presented to the network. When the error between the network output and target (desired output) is minimized, the network can be used in a testing

phase with test data. At this stage, the neural network is described by the optimal weight configuration, which means that theoretically ensures the output error minimization.

3.2 Classification-Based Test Case Prioritization

Clustering-based prioritization approach ranks test cases, complete event sequences(CESs), according to preference degrees as given by Eq (1)

$$PrefD(CES_q) = \sum_{i=1}^{n} Imp(x_i)\mu_{S_k}(x_i)f_q(x_i) \quad (1)$$

where $\mu_{S_k}(x_i)$ is the membership degree of the i^{th} event belonging to the cluster S_k , and $f_q(x_i)$ is the frequency of occurence of event i^{th} within CES_q. Note that $\mu_{S_k}(x_i)$ ranges between [0,1] in fuzzy clustering or its value is dichotomous {0,1} in hard clustering. $Imp(x_i)$ is the importance index of i^{th} event belonging to k^{th} cluster in defined as Eq (2):

$$Imp(x_i) = c - ImpD(S_k) + 1 \quad (2)$$

where *c* is the number of cluster, $ImpD(S_k)$ is the importance degree of the k^{th} cluster obtained by sorting the clusters in the decreasing order of the associated mean vectors, so that the cluster with the highest value has importance degree of 1, implying that it is the most important cluster of events among all [10].

The priorities of the test cases are determined by ranking the calculated $PrefD(CES_q)$ values from highest to lowest. This approach is only used to prioritize the test cases in the current test suite. If it is needed to add new test case to a test suite then this approaches less effective.

In order to overcome this drawback paper, clustering-based prioritization approach is improved by classifying the preference degrees of all existing test cases to obtain priority group label of any new test cases as well.

4. CASE STUDY

4.1. Classification with Multi-layer Perceptron NN

In this study, we selected 100 test cases derived from ESG of web-based software system. These test cases were prioritized by using our mentioned fuzzy clustering based TCP algorithm in advance. Then, ranking preference degrees of all test cases we determined five priority groups for the test suite. The five types of TCP group labels are as: very high priority, high priority, middle priority, low priority, very low priority. Test cases consist in totally 95 events number and ordering of which may be different in them. As seen from Eq (1),one of the attributes for preference degree is the membership degree$\mu_{S_k}(x_i)$each event has two attributes: importance index of events ($Imp(x_i)$) multiplied by membership degree and frequency of occurrence of event within a given test case ($f_q(x_i)$).

After definition of input and target data set d, the multi-layer perceptron network was constructed using MATLAB neural network tool. Constructed network properties are given in Table 1. Input layer of the network has 190 (95 x 2) neurons where each neuron represents the one attribute of the given event. Output layer of the network has 5 neurons which represent the priority groups. The Levenberg-Marquardt training function and the mean square error performance function were

used. The gradient descent with momentum weight/bias (*learngdm*) learning function was used as adaption learning function. Default values of l*earngdm*'s learning parameters: learning rate is 0.01 and momentum constant is 0.9. After several trying experiments, the number of hidden neuron in the network architecture was selected of 40. Transfer functions convert a neural networks layer's net input into its net output. In this study Log-sigmoid transfer function were used.

Table 1.Multi-layer Perceptron Network Components

Network Components	**Name & Values**
Network type	Feed-forward back propagation
Input data	190x100
Target data	5x100
Training function	Levenberg-Marquardt
Adaption learning function	learngdm
Performance function	Mean Square Error
Hidden layer node number	40
Transfer function	Logsig

Whole data set was randomly divided in training (70%), validation (15%) and test (15%) subsets. The network structure and training results are given in Figure1.

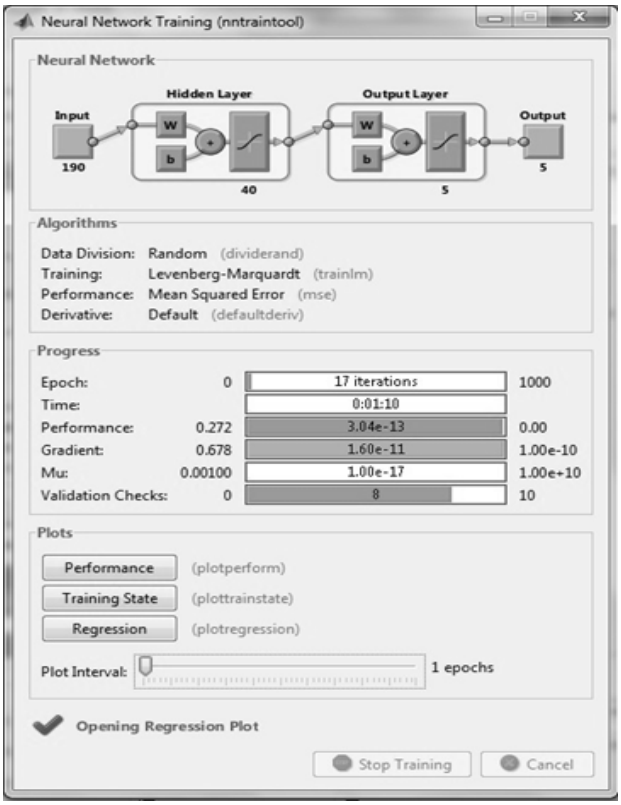

Figure 1.Screenshot of the result of the neural network training tool, MATLAB.

4.2 Experimental Results

At each time of training of a neural network can be obtain the different results due to random initial weight and bias values, and varying training, validation, and test data sets. As a result, neural network after training can give different outputs for the same input. In order to ensure getting a good accuracy of neural network it should be retrained for several times. In this study, we use the mean square error (MSE) to verify the performance of the classification. When the training was completed, we checked the network performance. Obtained results are given in Table 2.

Table 2. The results of five different training processes

	Training (R)	**Validation (R)**	**Test (R)**	**All (R)**	**Acc. %**		**Epoch**	**BVP (MSE)**	**BVP epoch**	**Time (sec)**
T1	0,99116	0,78967	0,83064	0,93927	96		14	0,064032	4	62
T2	0,96669	0,54498	0,85507	0,89211	92		17	0,12247	7	81
T3	0,99999	0,78778	0,83202	0,94491	96		20	0,063529	13	87
T4	0,99999	0,92653	0,92848	0,97871	96		17	0,022851	9	70
T5	1	1	0,9141	0,98726	99		24	1,876e-08	24	122

The first column from T1 to T5 in Table 2, represent the results of each training process. The network was trained with the parameters given in Table 1 in several times, and from the only best five trying experiments were selected. In this study, we used the mean square error (MSE) to verify accuracy performance of NN. After training, network performance results obtained are given in Table 2. In the left side of the table regression values (R) are given for training, validation and testing subsets, also all data set. The classification accuracy (Acc.%) is obtained with division of the number correctly classified test cases by total number of ones, (namely, 100) and accuracy values that ranged from 92% to 99% for each training process are given in Table 2. In the right side of the Table 2, the values of epoch, best validation performance (BVP), BVP epoch and total training times are given. Epoch represents maximum number of training iterations. BVP corresponds to the minimum MSE value, and BVP epoch corresponds to the number of iteration with minimum MSE. Finally, all training period is represented by time as second.

In Fig. 2 MSE values are given during the T4 training process as an example. As illustrated the figure, the validation performance reached a minimum was at 9 epoch. The training continued for 8 more iteration before the training stopped.

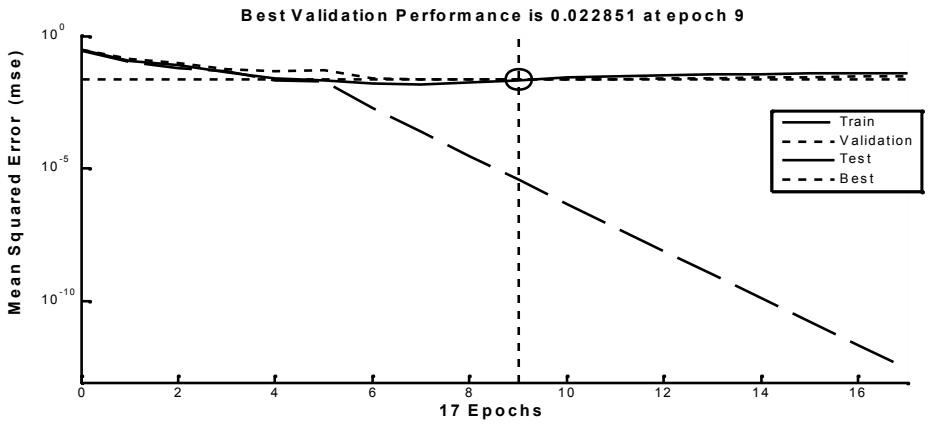

Figure 2.Best Validation Performance(BVP)

As seen from Figure2, there is no any major problems with the training NN. i.e. the validation and test curves are very similar and regression plots show the relationship between the values of network output and the targets. If the training were perfect, the network outputs and targets would be exactly equal, but the relationship is rarely perfect in practice.

The regression results are given in Figure3. There the four axes represent the training, validation, testing and all data. The dash line in each axis represents the perfect result. The solid line represents the best fit linear regression line between outputs and targets. R value represents the relationship between the outputs and targets. R=1, corresponds to exact linear relationship between outputs and targets.

As shown in Figure 3, NN for the training data set gets a very good fitting. In the case of testing of the validation, test and all data sets are obtained no poor results also where R values are greater than 0.926.

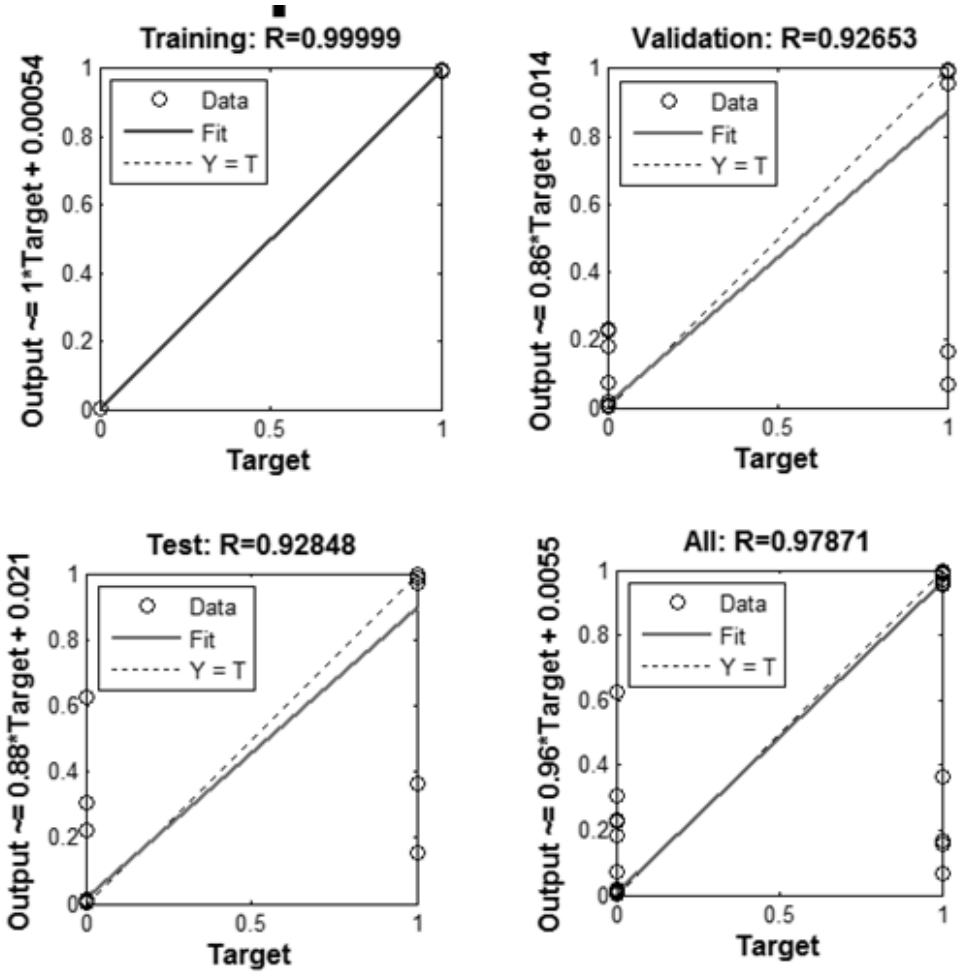

Figure 3.Regression plots

5. CONCLUSIONS

This paper deals with ESG model based test case prioritization problem for a large number of test cases. Improving our previous study, new model-event based TCP approach where instead of ordering indirectly test cases according to their preference degree they are automatically divided into the five groups(classes). It is provided thanks to representing prioritization group label of each test case as output depending on two attributes: important index weighted by membership degree and frequency of occurrence of all events belonging to given group. Then, such a way for all test cases (100) formed data set is classified by using MLP neural network. The structure of NN-classifier for commercial test application has been defined and its performance is examined. Application results show high classification accuracy and the acceptable test prioritization performance.

Suggested approach have obtained improving the effectiveness of TCP process in terms of the testing time and the capability to prediction of the group label for any new test cases added to the test suit afterwards as wells.

REFERENCES

[1] B. Beizer, “Software Testing Techniques. Second Edition”, New York, Van Nostrand Reinhold, 1990.

[2] S. Yoo and M. Harman, “Regression testing minimization, selection and Prioritization: A survey.” Software Testing, Verification and Reliability, Vol. 22, No. 2, 2012, pp. 67-120.

[3] G. Rothermel, R.H. Untch, C. Chu, M.J. Harrold, “Prioritizing Test Cases for Regression Testing”, IEEE Trans on Softw Eng. Vol.27, 2001 pp. 929-948.

[4] C. Catal and D. Mishra, “test case prioritization: a systematic mapping study”, Software Qual J, Vol. 21, 2013, pp. 445-478.

[5] H. Do,S. Mirarab, L: Tahvaldari and G. Rothermel, “The effects of time constraints on test case prioritization: A series of controlled experiments.”” IEEE Transaction on Software Engineering SE-2, Vol. 36, No. 5, 2010, pp. 593-617.

[6] B. Korel and G. Koutsogiannakis, “Experimental Comparison of Code-Based and Model-Based Test Prioritization”, In: Proc ICSTW’09, Denver, Colorado, 2009, pp. 77-84.

[7] F. Belli, “Finite-State Testing and analysis of Graphical User Interfaces”, In: Proc ISSRE’01, Hong Kong, China, 2001, pp. 34-43.

[8] N. Gökçe, M. Eminov and F. Belli, “Coverage-Based, Prioritized testing Using Neural Network Clustering”, The 21st International Symposium on Computer and Information Sciences (ISCIS 2006), November 1-3, İstanbul, Turkey, LNCS, 4263: pp. 1060-1071.

[9] N. Gökçe, F. Belli, M. Eminli, B. T. Dinçer, "Model-based test case prioritization using cluster analysis-a soft-computing approach", Turkish Journal of Electrical Engineering & Computer Sciences, DOI: 10.3906/elk-1209-109, 2013, (accepted).

[10] N. Gökçe, “Determination of Model Based Testing Priorities By Clustering Approach”, Doctoral Thesis (In Turkish), Mugla, Turkey, 2012.

[11] W.E. Wong, J.R. Horgan, S. London, A.P. Mathur, “Effect of Test Set Minimization on Fault Detection Effectiveness”, In: Proc. ICSE’95, Seattle, Washington, 1995, pp. 41-50.

[12] G. Rothermel, M.J. Harrold, J.V. Ronne, C. Hong, “Empirical Studies of Test-Suite Reduction”, J Softw Test Verify, and Reliab. Vol.12, pp. 219-249, 2002.

[13] P.R. Srivastava, “Test Case Prioritization”, J of Theor and ApplInfTechnolog, pp. 178-

[14] W. Wong, J. Horgan, S. London, H. Agrawal, “A study of effective regression testing in practice”, In Proc. ISSRE 1997, Albuquerque, New Mexico, pp. 230-238, 1997.

[15] J.M. Kim, A. Porter, „A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments”, In: Proc ICSE’02, Orlando, FL, pp. 119-129, 2002.

[16] H. Srikanth, L. Williams, J. Osborne, “System Test Case Prioritization of New and Regression Tests”, In: Proc ISESE 2005, Noosa Heads, Australia, pp. 1-10, 2005.

[17] R. Krishnamoorthi, S.A. Sahaaya Arul Mary, “Requirement Based System Test Case Prioritization of New and Regression Test Cases”, Int J of SoftwEng and Knowl Eng. Vol.19, pp. 453-475, 2009.

[18] A. Srivastava, J. Thiagrajan, “Effectively Prioritizing Test in Development Environment”, In: Proc ISSTA2002, Roma, Italy, pp. 97-106, 2002.

[19] S. Sabharwal, R. Sibal and C. Sharma, “Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from UML Diagrams”, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011, pp. 433-444.

[20] F. Belli, M. Eminov, N. Gökçe, “Coverage-Oriented, Prioritized Testing – A Fuzzy Clustering Approach and Case Study”, Dependable Computing, LNCS Vol. 4746, pp. 95-110, 2007b.

[21] D.E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning representation by backpropagation errors”, Nature, Vol.323, No.6088, 1986, pp.533-536

[22] L.M. Silva, J. Marques de Sa and L.A. Alexandre, “Data classification with multilayer perceptrons using a generalized error function”, Neural Networks, Vol. 21, 2008, pp. 1302-1310.

AUTHORS

NIDA GÖKÇE was born in 1976. She received the B.S. degree in 2001 and M.S. degree in 2006 in Statistics and Computer Science from Muğla University. She had been a scholarship holder in the Department of Software Engineering at Paderborn University during 2009-2010. She completed her Ph.D. in determination of model based test priorities by clustering approach at Muğla University in 2012. Since 2002 she has been working at the same university, in the Department of Statistics. Her research interest includes model based software testing, artificial neural networks, soft computing and clustering.

MÜBARIZ EMİNLİ was born in 1948. He received the B.S. and M.S. degrees in computer engineering from Azerbaijan Technical University in 1971, and Ph.D. degree from Azerbaijan Pertol and Chemical Academy in 1987. From 1974 to 1996 he worked as a researcher at the Institute of Cybernetics in the Azerbaijan National Academy of Science. From 1996 to 2008 he has worked in Mugla University, Turkey. Since 2008, he has been working in Haliç University (Istanbul) at the Department of Computer Engineering. His research interest includes fuzzy clustering, fuzzy modeling, neural network and data mining.