HOSTED BY
ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

International Journal of e-Navigation and Maritime Economy 2 (2015) 47 – 62

International Journal of e-Navigation and Maritime Economy

www.elsevier.com/locate/enavi

Original article

Performance Verification Mechanism for Adaptive Assessment e-Platform and e-Navigation Application*

Chang-Shing LEE†, Mei-Hui WANG, Cheng-Hao HUANG

† Dept. of Computer Science and Information Engineering, National University of Tainan, Taiwan
leecs@mail.nutn.edu.tw, Corresponding Author

Abstract

Adaptive assessment e-platform is being promoted in the world to make teachers understand students' e-learning performance on the Internet. However, system's load testing for an adaptive assessment is a very important issue during development of such an e-platform. In this paper, we have adopted the genetic fuzzy markup language (GFML) to infer the performance of an adaptive assessment e-platform. Firstly, we collected the data and information of the e-platform loading in two different mechanisms. With the collected data, the proposed CPU usage calculation mechanism is first implemented to acquire the CPU usage information from the screenshot of Ganglia. Next, we used the fuzzy c-means (FCM) clustering mechanism to construct the knowledge base according to the collected data. Then, number of threads, constant timer, MySQL parameter, CPU usage, and testing time of the e-platform were utilized to infer the e-platform load performance. Finally, the genetic learning algorithm was utilized to learn the knowledge and rule base to optimize the proposed approach. From these experimental results, the proposed method is feasible for verifying the performance of an adaptive assessment e-platform. In the future, the adaptive assessment e-platform can be utilized to e-Navigation systems and applications.

Keywords: Genetic Fuzzy Markup Language, Item Response Theory, Fuzzy Inference, Genetic Algorithm, Adaptive Assessment e-Platform

* This paper is a revised version presented at the 2nd Ai-MAST held at Mokpo National Maritime University in Mokpo, Korea, May 17-19, 2014.

I. Introduction

Item response theory (IRT) is a psychometric paradigm for the construction, scoring, and analysis of test forms and items (Thompson, 2009). IRT considers that there exists a certain kind of relationship between the performance of students with a different ability and an item. This relationship is called an item characteristic curve (ICC) to represent the possible relationship between students' ability (or a certain latent trait) and probability that students correctly answer this item (Yu, 2009; Embretson and Reise, 2000). Owing to the advances in computing and information technology, computerized adaptive testing (CAT) is being introduced to the world. For example, Programme for International Student Assessment (PISA) defines increasing the use of adaptive testing to be one of the six objectives for the longer-term development of PISA (PISA, 2014). The main concept of CAT is to choose the next item for an examinee based on his/her last-item response (Ree, 1997). When an examinee correctly answers the last item, CAT chooses a more difficult item to be his/her next item. On the contrary, CAT chooses an easier item for the next one when he/she makes a wrong response to the last item.

Currently, students no longer have to passively consume learning materials but have to actively create and disseminate knowledge (Munoz-Organero et al., 2010) so that there is some research regarding the application of computational intelligence to adaptive learning. For example, Lee et al. (2014a) presented a type-2 fuzzy set (T2 FS)-based adaptive linguistic assessment system to evaluate human performance on the game of Go. Melia and Pahl (2009) proposed a courseware authoring validation information architecture to adaptively validate courseware. Munoz-Organero et al. (2010) proposed a service-oriented personalized e-learning environment to make users consume external e-learning services orchestrated by an IMS-LD. Sampayo-Vargas et al. (2013) used an educational computer game to evaluate the effectiveness of adaptive difficulty adjustments on students' motivation and learning. Fuzzy markup language (FML) is an XML-based language for enabling full interoperability in fuzzy systems design (Acampora and Loia, 2005; Acampora et al., 2013). FML and genetic FML (GFML) allow a fuzzy reasoning to be distributed and applied to many applications like ambient intelligence (Acampora and Loia, 2008), NoGo (Lee et al., 2012b), computer Go (Lee et al., 2013; Lee et al., 2014a), dietary assessment (Lee et al., 2012a; Lee et al., 2014c), and so on. Additionally, the IEEE Computational Intelligence Society (CIS) P1855 Working Group members are enacting the documents to make FML the first standard technology in the field of computational intelligence.

One of the most popular clustering algorithms is fuzzy c-means (FCM) clustering mechanism and it was proposed by Bezdek in 1981 (Bezdek, 1981; Cannon et al., 1986; Bezdek et al., 1984). FCM-related applications to problems in clustering, feature selection, and classifier design have been reported in very large data (Havens et al. 2012) and other domain areas. For example, Zarinbal et al. (2014) presented an interval type-2 relative entropy fuzzy c-means clustering and

showed that it has a very good ability to detect noises. Ji et al. (2014) developed the interval-valued possibilistic fuzzy c-means clustering algorithm to segment the brain magnetic resonance images and natural images. Wikaisuksakul (2014) presented a multi-objective genetic algorithm with fuzzy c-means for automatic data clustering. Genetic algorithm (GA) is a popular approach to optimizing the performance according to search heuristic (Chen et al., 2013). Additionally, the use of genetic fuzzy systems has been widely extended because of their inherent flexibility and their capability to jointly consider different optimization criteria (Cordon, 2011). Until now, there has been a lot of research on GA. For example, Lee et al. (2012a) and Lee et al. (2012b) utilized a novel genetic fuzzy markup language and applied it to healthy diet assessment and game of NoGo, respectively. Meng and Pei (2012) combined fuzzy logic and GA to extract linguistic rules from data sets. Ghanbari et al. (2013) developed a Cooperative Ant Colony Optimization-Genetic Algorithm (COR-ACO-GA) to construct energy demand forecasting knowledge-based expert systems. Hong et al. (2014) used an effective parallel approach for genetic-fuzzy data mining to overcome the low-speed fitness evaluation problem of the original algorithm.

Owing to the promotion of the adaptive assessment e-platform in the world, the developed system's load testing for an adaptive assessment e-platform becomes an important issue during development of such an e-platform. Hence, we propose the performance verification mechanism for the adaptive assessment e-platform and e-navigation application. The technical contents in terms of innovation and significance are as follows: The key contribution of this paper is to use the genetic fuzzy markup language (GFML) to describe the knowledge base and rule base of the e-platform performance verification for elementary-school and junior high-school students' assessment e-platform in Taiwan. We collect the data and information of e-platform loading in two different mechanisms. One is to use JMeter to simulate lots of users to do the test within a specific period of time and make a response to all selected items. Another is there are about 10,000 students surfing on the e-platform to do adaptive testing. Next, we use the *fuzzy c-means* (FCM) *clustering mechanism* to construct the fuzzy sets according to the collected data. And, the *weight-based fuzzy rule construction mechanism* is executed to construct the fuzzy rules. After domain expert's verification and validation, the knowledge and rule base of the performance verification and validation mechanism are built. Finally, the system load is inferred to evaluate the e-platform performance. The remainder of this paper is as follows: Section II briefly introduces the Program of Learning Diagnosis and Progress Assessment (POLDPA) of Kaohsiung project in Taiwan. The e-platform performance verification mechanism is presented in Section III. Experimental results are given in Section IV. Finally, conclusions are given in Section V.

II. Program of Learning Diagnosis and Progress Assessment (POLDPA) of Kaohsiung in Taiwan

2.1. POLDPA Routing Architecture Overview

In this paper, the system load testing for the POLDPA e-platform focuses on the adaptive e-testing subsystem. Figure 1 shows the routing architecture diagram for the POLDPA e-platform. We give some brief descriptions of Figure 1 as follows: (1) Junior high-school or elementary-school students of Kaohsiung connect to the Internet via their computers at the computer classroom; (2) After connecting to Education Bureau of Kaohsiung city government, students surf on the POLDPA e-platform which is located at the National Center for High-Performance Computing (NCHC), Taiwan; (3) After passing the account's authentication, POLDPA's adaptive e-testing subsystem is activated by judging students' response to the last item, estimating students' ability based on their response to the item, and selecting the item that suits students' current ability to be the next item; (4) Students continue to make a response to the selected item until (a) maximum number of items has been responded or (b) minimum number of items has been responded and the minimum standard deviation error has been reached (Lee et al., 2014b).

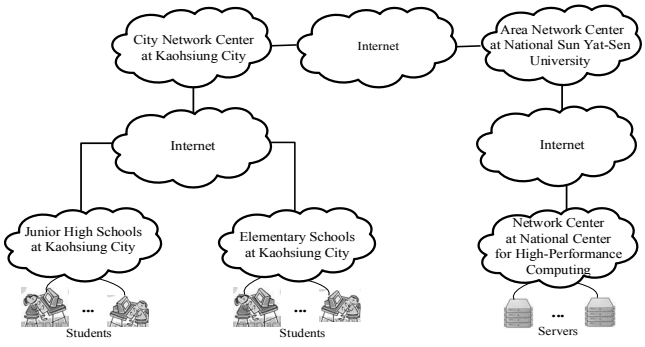


Figure 1: Routing architecture diagram for POLDPA e-platform (Lee et al., 2014b)

2.2. Introduction to System Load Testing Structure for POLDPA e-Platform

System load testing is one of the methods used to test the limits of the performance of the servers, network, and algorithm in order to find the optimized performance of the system. Figure 2 shows the system test structure for POLDPA e-platform to verify the e-platform's performance. The brief description is as follows: (1) Have a meeting with domain experts for testing plan; (2) Follow the principle of software engineering to design our test case and test plan (TP). Each TP includes this test-case name, code, environmental setting, testing schedule, job allocation, test condition, test input, test steps, expected results, criteria for pass and failure, terminated condition, and so on; (3) **For simulation data collection:** (a) Use JMeter to record each TP's scenario to generate scripts for all TPs. (b) Based on the generated scripts, operators use PCs No. *1*, No. *2*, …,

and No. *N* to send specific threads to stimulate lots of students to surf on the Servers No. *1*, No. *2*, …, No. *M*, to do adaptive testing. Meanwhile, operators observe the real-time resource usage condition like *system load averages*, *memory usage averages*, *CPU utilization averages*, *network bandwidth usage averages*, and so on, via Ganglia, which was developed by Computer Science Division, University of California, Berkeley, USA. (c) After finishing the simulation, operators store the collected data from JMeter and Ganglia onto the data repository; (4) **For actual data collection:** (a) Teachers advise students to follow the test plan to surf on the POLDPA e-platform to do adaptive testing. (b) After students finished the testing, the collected data from the POLDPA servers and Ganglia are stored onto the data repository.

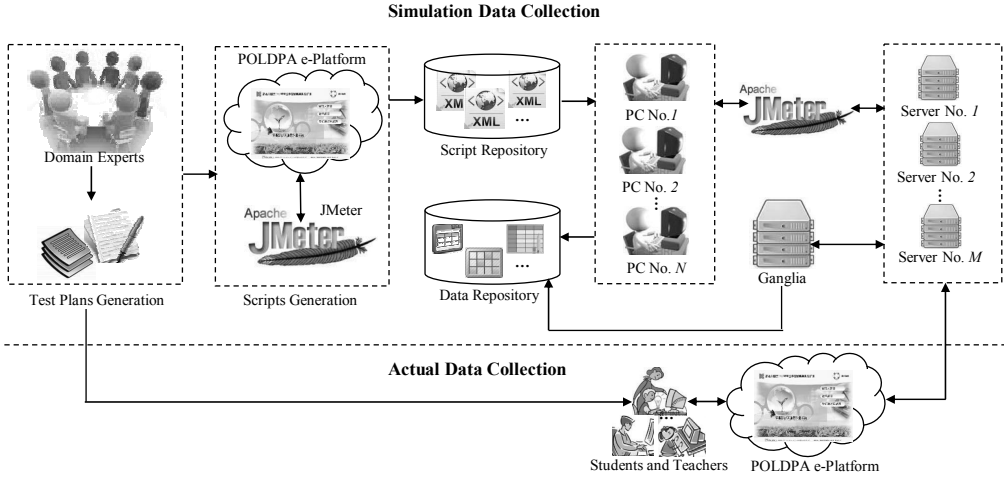


Figure 2: System load testing structure for POLDPA e-platform (Lee et al., 2014b).

2.3. System Structure of Performance Verification and Validation for e-Platform

Figure 3 shows the adopted system structure for POLDPA e-platform performance verification and validation, including a *CPU usage mechanism*, a *FCM-based fuzzy set construction mechanism*, a *weight-based fuzzy rule construction mechanism*, a *fuzzy inference mechanism*, and a *genetic learning mechanism*. Herein, the fuzzy inference mechanism is composed of a fuzzifier, an inference, and a defuzzifier. The inputs are *number of threads* (*NT*), *constant timer* (*CT*), *MySQL parameter* (*MP*), *CPU usage* (*CU*), and *testing time (TT*). Based on the pre-defined knowledge base (KB) and rule base (RB), the *system load* (*SL*) is inferred after implementing the fuzzy inference mechanism.

The reasons that we choose *NT*, *CT*, *MP*, *CU*, and *TT* to input are as follows: (1) In our system load testing, we use JMeter to simulate users to connect to the server. Therefore, *number of threads* decide how many threads are issued by all of the available PCs; (2) *Constant timer*

denotes one thread pause for the same amount of time between requests when we use JMeter to simulate the users. For simulation data, we usually set the constant timer to be 1 second to simulate from the one item that responds within about 1 second. However, for actual students' test, one item usually responds about between 30 seconds to 50 seconds; (3) Tuning parameters for relational database management system (RDMS) and HTTP server is an important job to optimize the server's performance; (4) *CPU usage* derives from the CPU utilization across all processes on all systems of Ganglia. Based on CPU last-hour information captured from Ganglia, we can acquire the CPU usage during testing; (5) POLDPA e-platform frequently queries and stores data via a database during testing in order to choose the next item that fits current students' estimated ability. If such a kind of I/O operation fails, the testing time of one completed test will shorten. Hence, the length of the *testing time* is an indicator to show whether this testing is successful or has failed.

Because of the above-mentioned explanations, *NT*, *CT*, *MP*, *CU*, and *TT* are chosen as our input fuzzy variables to infer the *SL*. Herein, *system load* represents the number of users that successfully finish the testing. Additionally, we also executed the genetic learning mechanism to optimize the knowledge base and rule base based on the selected training data to improve the performance of the proposed approach.

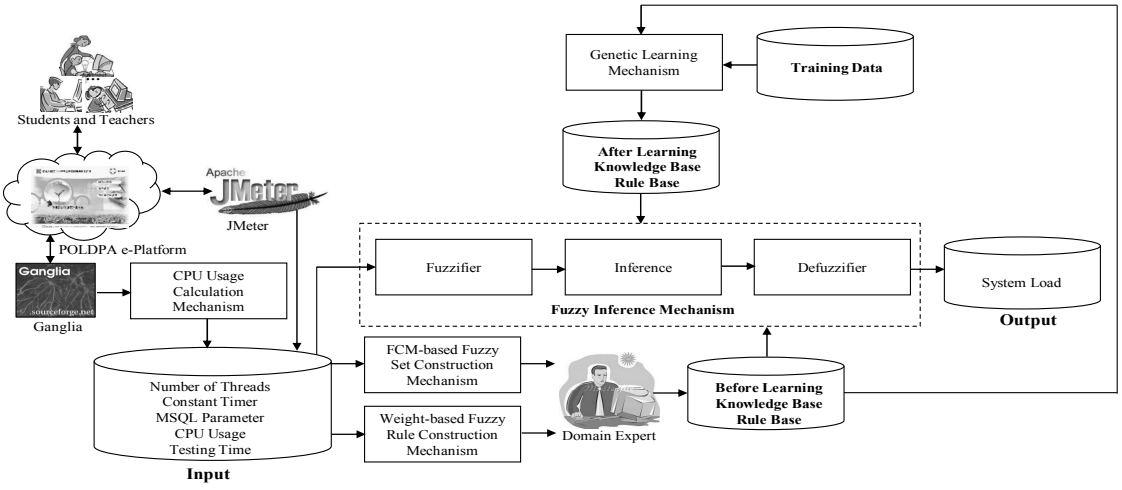


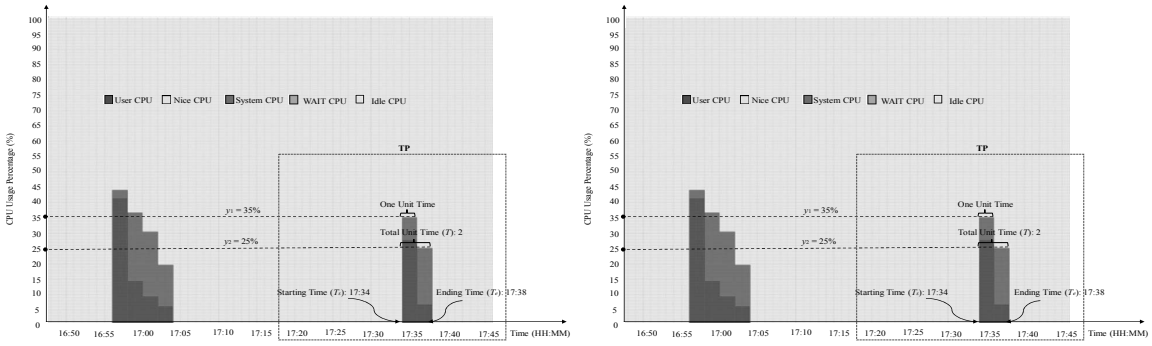
Figure 3: System structure for e-platform performance verification and validation (Lee et al., 2014b).

III. E-platform Performance Verification Mechanism

3.1. CPU Usage Calculation Mechanism

This subsection introduces how to acquire CPU usage from the collected data. *CPU usage* from Ganglia, including percentages of *User CPU*, *Nice CPU*, *System CPU*, *WAIT*

CPU, and *Idle CPU*, is a good indicator to see whether the server works efficiently.



(a) (b)

Figure 4: CPU usage percentage for one TP with a (a) shorter and (b) longer testing time (Lee et al., 2014b).

Figure 4 is a captured CPU usage image from Ganglia when we executed one test plan on the afternoon of April 19, 2014, and this TP started at 17:34 and ended at 17:38. Additionally, Figure 4(a) also shows that total number of unit time (T) is 2 whose starting time (T_s) and ending time (T_e) are 17:34 and 17:38, respectively. The values of input fuzzy variables *CU* and *TT* are calculated by Eqs. 1-2, respectively. Therefore, the values of *CU* and *TT* are 30% and 4 minutes, respectively, according to Figure 4(a). Another example for one TP with a good performance is shown in Figure 4(b). Observe that Figure 4(b) has a bigger area than Figure 4(a) does during the testing time.

$$CU(\%) = \frac{\sum_{i=1}^{T} y_i}{T} \tag{1}$$

where, T denotes total number of unit time between starting time (T_s) and ending time (T_e), and y_i denotes the CPU usage percentage for the i^{th} unit time.

$$TT = T_s - T_e \tag{2}$$

3.2. FCM-based Fuzzy Set Construction Mechanism

In this paper, we use FCM to construct the initial fuzzy sets. FCM is based on minimization of the object function, shown in Eq. 3. The FCM algorithm, via an iterative optimization of J_m, produces a fuzzy c partition of the data set $X=\{x_1, x_2, .., x_n\}$. Fuzzy partitioning is carried out through an iterative optimization of Eq. 3 with the update of membership u_{ij} (calculated by Eq. 4), and the cluster centers c_j (calculated by Eq. 5). The iteration will stop when Eq. 6 is met (DEIB, Politecnico di Milano, Italy, 2014; Cannon et al., 1986). Based on the clustering results and domain expert's knowledge, we construct the fuzzy sets for the

adopted fuzzy variables. Figures 5(a)-5(f) show the adopted fuzzy sets for fuzzy variables *NT*, *CT*, *MP*, *CU*, *TT*, and *SL*, respectively. Each linguistic (fuzzy) variable has an associated fuzzy term set. For example, *NT* has three fuzzy term sets, including *Low*, *Medium*, and *High*.

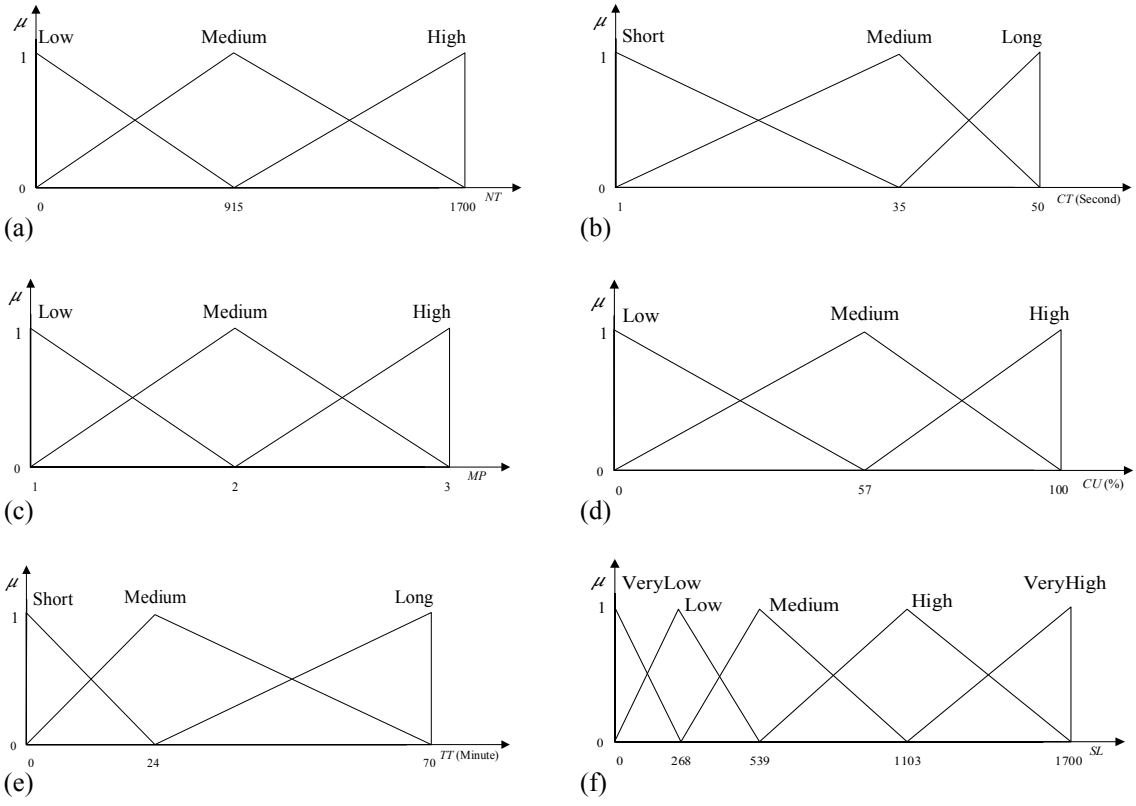


Figure 5: Fuzzy sets for fuzzy variables (a) *NT*, (b) *CT*, (c) *MP*, (d) *CU*, (e) *TT*, and (f) *SL*

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^m \left\| x_i - c_j \right\|^2, \; 1 \le m < \infty, \; 1 \le c < n \tag{3}$$

where, m is any real number greater than 1, n is the number of data items, u_{ij} is the membership degree of x_i in the cluster j, x_i is the i^{th} of d-dimensional measured data, c_j is the d-dimension center of the cluster, and $\|*\|$ is any norm expressing the similarity between any measured data and the center (DEIB, Politecnico di Milano, Italy, 2014; Cannon et al., 1986).

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_k \right\|} \right)^{\frac{2}{m-1}}} \tag{4}$$

$$c_j = \frac{\sum_{i=1}^{n} u_{ij}^m x_i}{\sum_{i=1}^{n} u_{ij}^m} \tag{5}$$

$$\left\| U^{(k+1)} - U^k \right\| < \varepsilon \tag{6}$$

where $U^{(k+1)}$ is the fuzzy c partition at step k+1, U^k is the fuzzy c partition at step k, and ε is a termination criterion between 0 and 1.

3.3. Genetic Learning Mechanism

The adopted genetic learning mechanism (Lee et al., 2012a) operates as follows: (1) The initialization mechanism uses the encoded before-learning GFML to be the original chromosome, and then mutates it until the size of population is reached; (2) The chromosomes in the population are evaluated by executing the evaluation mechanism; (3) Parents are selected by the roulette-wheel selection mechanism, which is one of the most common techniques being used for a proportionate selection mechanism; (4) The single-point crossover and mutation mechanism executes the crossover and mutation for the parents to generate their offspring based on the crossover rate and mutation rate, and they are evaluated by the evaluation mechanism; (5) If the fitness value of the generated offspring is better than the worst chromosome in the population, then the offspring replaces the worst one by executing the replacement mechanism. In this paper, we use mean square error (*MSE*) as the fitness function, calculated by Eq. 7. If the fitness value of the best chromosome (*Fitness*) satisfies the minimum error ($error_{min}$), then stop the evolution. Otherwise, the genetic learning mechanism continues to evolve until the maximum number of generation (N_{GEN}) is reached; (6) Finally, the best chromosome is decoded into an after-learning GFML.

$$MSE = \frac{\sum_{i=1}^{N}(x_i - y_i)^2}{N} \tag{7}$$

where N denotes the number of training data, x_i and y_i are the inferred output and the desired output, respectively, for the i^{th} record of the training data.

IV. Experimental Results

4.1. Data Collection

This section introduces the experimental results. The input data are divided into two groups. One is simulation data from JMeter on April 19, April 29, and April 30, 2014. Another is the actual data collected from about 10,000 students' adaptive e-testing on May 26, 2014. Below is their brief descriptions:

- **For simulation data:**

– We use multi-computers installed JMeter to send multi-threads to simulate lots of students to do the adaptive e-testing via ten servers (M1~M10).

– Each script does the account authentication, confirms the tested subject, makes a response to the selected 25 items via I/O database, and writes the simulated students' evaluated ability and

percentile rank (PR) into the database.

- The following variables are variable to simulate different kinds of simulation conditions: (1) set the value of the *number of threads* (users) for each computer to simulate how many users are simultaneously surfing the e-platform to do the test, (2) set *constant timer*'s thread delay to introduce how many mini-seconds are delayed between consecutive requests of the same thread, and (3) set the *MySQL parameter*, like *connect_timout*, *max_connections*, *max_connect_errors*, *query_cache_limit*, and so on, to check if *CPU usage* of the server could be increased.
- After finishing testing, the testing time and some data are collected and then they are stored in the database. Below is the brief descriptions of the collected data: (1) SL_A: *Number of threads* that successfully pass the authentication; (2) SL_B: *Number of threads* that are with the estimated ability and PR after testing; (3) SL_C: *Number of threads* that are with the valid estimated ability and PR after testing; (4) SL_D: *Number of threads* that are with the valid estimated ability and PR. Additionally, these threads also make a successful response to all of the selected items (25 items); (5) SL_{DO}: An average of SL_A, SL_B, SL_C, and SL_D that is the desired output of the *system load* for the genetic learning mechanism. Table 1 shows the collected data at NCHC on April 29, 2014.

- **For actual data:**

About 10,000 students joined this system load testing on May 26, 2014. In order to reduce the instantaneous system load, we set up one entrance server to dispatch the system load into eight servers (M1~M6, M9, and M10) according to the administrative district which the involved students' schools are located. We use the expected the number of involved students as the *number of threads*. Additionally, the average response time for one item is regarded as the *constant timer*. After finishing testing, the *testing time*, SL_A, SL_B, SL_C, and SL_D, and SL_{DO} are stored into the database. Table 2 shows the actual data collected at Kaohsiung on May 26, 2014.

Table 1: Simulation data at NCHC on April 29, 2014.

Server	TP	*NT*	*CT* (Sec)	*MP*	*CU* (%)	*TT* (Min)	SL_{DO}
M1	TP131	1200	1	3	54.5	4	269.5
	TP141	1200	1	3	56	4	269.5
M2	TP132	1200	1	3	53.5	4	227.5
	TP142	1200	1	3	49.5	4	267.25
M3	TP133	1200	1	3	47.5	4	223
	TP143	1200	1	3	45.5	4	289
M4	TP134	1200	1	3	62.5	4	359.75
	TP144	1200	1	3	43.5	4	296
M5	TP135	1200	1	3	34.6	5	275.5

M6	TP136	1200	1	3	36.3	6	301.5
M7	TP137	1200	1	3	42	4	222.25
M8	TP138	1200	1	3	42	4	213.5
M9	TP139	1200	1	3	77.5	20	1192
M10	TP140	1200	1	3	80.5	20	1200

Table 2: Actual data collected at Kaohsiung on May 26, 2014.

Server	TP	*NT*	*CT* (Second)	*MP*	*CU* (%)	*TT* (Minute)	SL_{DO}
M1	TP-M1	1680	46	3	4.55	46	1166.25
M2	TP-M2	1620	43	3	7.95	46	1159.5
M3	TP-M3	1680	33	3	16.86	46	815
M4	TP-M4	1680	34	3	9.85	45	998.75
M5	TP-M5	1560	44	3	52.73	67	1211.75
M6	TP-M6	1500	42	3	9.61	36	953
M9	TP-M9	1470	45	3	8.04	44	1054.75
M10	TP-M10	1500	45	3	8.58	48	979.25

4.2. Genetic Learning Performance Comparison between before Learning and after Learning

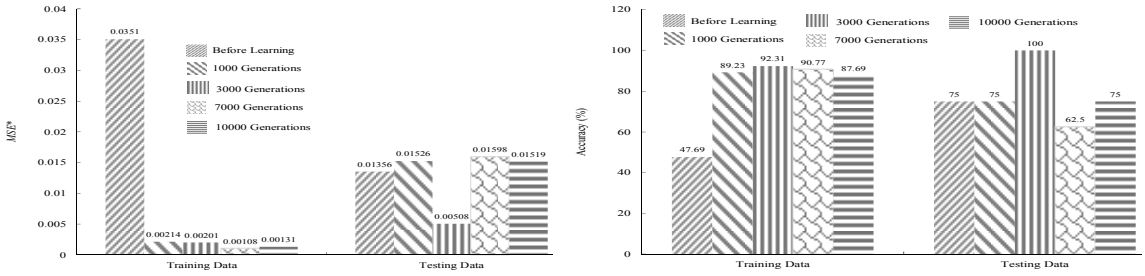
We first use 16 pairs of crossover rate and mutation rate (*CR* / *MR*) to do some experiments in learning the knowledge base and rule base of the adaptive assessment e-platform load verification for 3000 generations. The total number of collected data is 73, where 65 data are used for training and 8 data are used for testing. That is, all of the simulation data are for training and the actual data are for testing. In addition, we use *MSE* as a criteria in order to evaluate the performance of the genetic learning. However, in order to make the range of *MSE* to be [0, 1], MSE^* is calculated by Eq. 8. Additionally, the accuracy criteria, calculated by Eq. 9, is also adopted in this paper. Observe that for training data, 0.9/0.05 has the best performance after learning. For testing data, 0.9/0.1 has the best performance. Because both 0.9/0.1 and 0.9/0.05 have the better performance, we use 0.9/0.1 and 0.9/0.05 to evolve for 1000, 7000, and 10000 generations. Figures 6(a) and 6(b) show the MSE^* and accuracy, respectively, before learning and after learning 1000, 3000, 7000, and 10000 generations when crossover rate is 0.9 and mutation rate is 0.1. From the observation of Figures 6(a) and 6(b), they indicate that evolving 3000 generations can acquire a much better performance for training data and testing data. Figures 7(a)-7(f) show the after-learning fuzzy sets for the adopted fuzzy variables *NT*, *CT*, *MP*, *CU*, *TT*, and *SL*, respectively, when generation is 10000, crossover rate is 0.9, and mutation rate is 0.1.

$$MSE^* = \frac{MSE}{SL_{DomainRight} - SL_{DomainLeft}} \tag{8}$$

where *MSE* is calculated by Eq. 7, as well as $SL_{DomainRight}$ and $SL_{DomainLeft}$ denote the domain right and domain left of the fuzzy set *SL*, respectively.

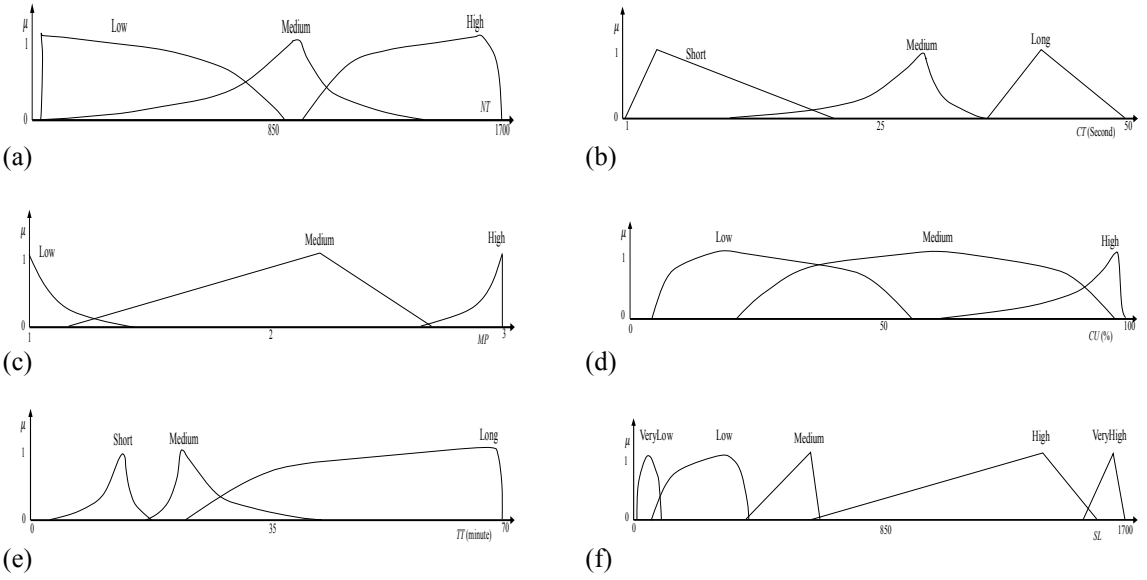
$$Accuracy\ (\%) = (x/y) \times 100 \tag{9}$$

where *x* denotes the number when the inferred linguistic description of *SL* exactly matches with the domain expert's and *y* denotes the number of experimental examples.



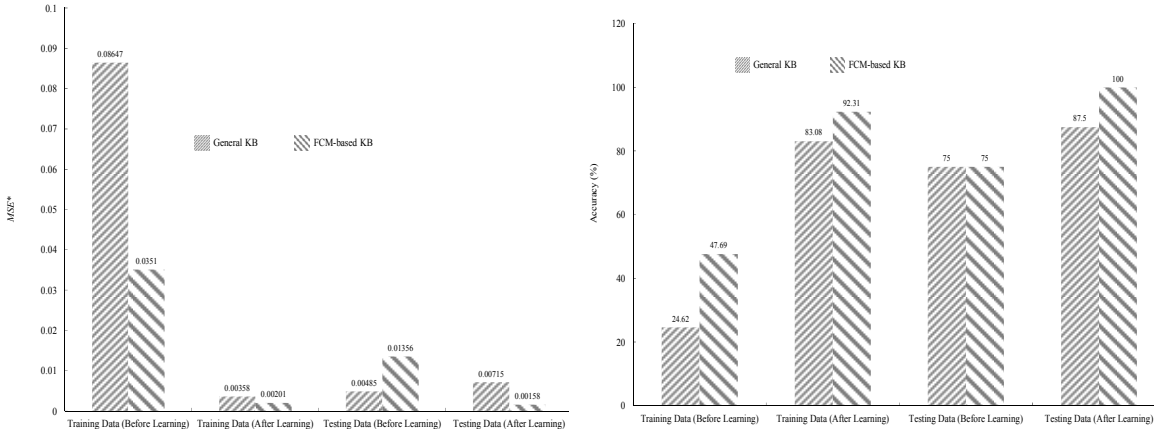
(a) (b)

Figure 6: (a) MSE^* and (b) accuracy before learning and after learning 1000, 3000, 7000, and 10000 generations when crossover rate is 0.9 and mutation rate is 0.1



(a) (b) (c) (d) (e) (f)

Figure 7: After-learning fuzzy sets for fuzzy variables (a) NT (b) CT, (c) MP, (d) CU, (e) TT, and (f) SL when generation is 10000, crossover rate is 0.9, and mutation rate is 0.1.



(a) (b)

Figure 8: (a) *MSE and (b) accuracy comparison between general KB and FCM-based KB**

4.3. Genetic Learning Performance based on FCM Clustering Mechanism

In this subsection, we want to validate the performance of *FCM clustering mechanism*. We compared the performance between general KB and FCM-based KB (see Figure 5). Figures 8(a) and 8(b) show the behaviour of *MSE** and accuracy when generation is 3000, crossover rate is 0.9, and mutation rate is 0.1, respectively. Observe that the FCM-based KB outperforms the general KB from the viewpoint of the *MSE** and accuracy.

V. Conclusion

In this paper, we infer the system load according to the *number of threads*, *constant timer*, *MySQL parameter*, *CPU usage*, and *testing time* of the system servers in the e-platform. Additionally, two kinds of data are collected to validate the performance of the proposed approach. We also used the genetic algorithm to learn the knowledge base and rule base to optimize the performance of the e-platform. Experimental results show that the proposed approach is feasible to apply to the adaptive assessment e-platform performance verification and validation. The practical applications of such a study is also possible to contribute to the performance verification of e-Navigation systems that will be developed in the near future. However, current performance still exhibits some weaknesses. Compared to the traditional approach, a type-2 fuzzy has its superiority, for example, a type-2 fuzzy set is three-dimensional and its additional dimension provides it to have additional degrees of freedom to model and handle the real-world uncertainty (Mendel, 2001). Hence, in the future, we will compare the experimental results and make

the discussions with an alternative method like type-2 fuzzy set or data mining to improve the performance of the proposed approach. Furthermore, the improved adaptive assessment e-platform also can be applied to other countries in Asia (e.g. Korea and Japan), Europe, America, etc.

Submitted: September 30, 2014 Accepted: December 5, 2014

Acknowledgements

The authors would like to thank the financial support by the Ministry of Science and Technology of Taiwan under the grant NSC 102-2221-E-024-005 & MOST 103-2221-E-024-008 and POLDPA. Additionally, the authors would like to thank members from OASE Lab. (Ontology Application & Software Engineering Laboratory, NUTN, Taiwan), KWS (Center for research of Knowledge application & Web Service, NUTN, Taiwan), NCHC, Education Bureau of Kaohsiung city government as well as Fu-Sheng Pai, Su-Wei Lin, Pi-Hsia Hung, Jee-Gong Chang, Ce-Kuen Shieh, Te-Ming Chen, and Chin-Hung Li for their kind help with the system load testing.

References

Acampora, G., Loia, V., Lee, C.S. and Wang, M.H. (2013), On the power of fuzzy markup language, Springer-Verlag, Germany.

Acampora, G. and Loia, V. (2005), Fuzzy control interoperability and scalability for adaptive domotic framework, IEEE Transactions on Industrial Informatics, Vol. 1, No. 2, pp. 97-111.

Acampora, G. and Loia, V. (2008), A proposal of ubiquitous fuzzy computing for ambient intelligence, Information Sciences, Vol. 178, No. 3, pp. 631-646.

Bezdek, J. (1981), Pattern recognition with fuzzy objective function algorithms, Kluwer Academic Publishers Norwell, MA, USA.

Bezdek, J.C., Ehrlich, R. and Full, W. (1984), FCM: The fuzzy c-means clustering algorithm, Computers & Geosciences, Vol. 10, No. 2-3, pp. 191-203.

Cannon, R.L., Dave, J.V. and Bezdek, J.C. (1986), Efficient implementation of the fuzzy c-means clustering algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 2, pp. 248-255.

Cordon, O. (2011), A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: designing interpretable genetic fuzzy systems, International Journal of Approximate Reasoning, Vo. 52, No. 6, pp. 894-913.

Chen, H., Jiang, W., Li, C. and Li, R. (2013), A heuristic feature selection approach for text categorization by using chaos optimization and genetic algorithm, Mathematical problems in Engineering, Vol. 2013, pp. 1-6.

DEIB, Politecnico di Milano, Italy (2014), A tutorial on clustering algorithms: fuzzy c-means clustering, website: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html.

Embretson, S.E. and Reise, S.P. (2000), Item Response Theory: Taylor & Francis.

Ghanbari, A., Kazemi, S.M.R., Mehmanpazir, F. and Nakhostin, M.M. (2013), A cooperative ant colony optimization-genetic algorithm approach for construction of energy demand forecasting knowledge-based expert systems, Knowledge-Based Systems, Vol. 39, pp. 194-206.

Havens, T.C., Bezdek, J.C., Leckie, C., Hall, L.O. and Palaniswami, M. (2012), Fuzzy c-means algorithms for very large data, IEEE Transactions on Fuzzy Systems, Vol. 20, No. 6, pp. 1130-1146.

Hong, T.P., Lee, Y.G. and Wu, M.T. (2014), An effective parallel approach for genetic-fuzzy data mining, Expert Systems with Applications, Vol. 41, No. 2, pp. 655-662.

Ji, Z., Xia, Y., Sun, Q. and Cao, G. (2014), Interval-valued possibilistic fuzzy c-means clustering algorithm, Fuzzy Sets and Systems, Vol. 253, pp. 138-156.

Lee, C.S., Wang, M.H., Wu, M.J, Teytaud O. and Yen, S.J. (2014a), T2FS-based adaptive linguistic assessment system for semantic analysis and human performance evaluation on game of Go, IEEE Transactions on Fuzzy Systems, Vol. 23, No. 2, pp. 400-419.

Lee, C.S., Wang M.H., Huang, C.H., Cho, S.L., Wang, C.S., Tsai, B.H., Yu, J.L., Chang, C.H., Hsieh, P.J., Chen, T.M. and Li, C.H. (2014b), FML-based website performance verification mechanism for adaptive assessment system, The 2nd International Symposium on Advanced Intelligent Maritime Safety and Technology (Ai-MAST 2014), Mokpo, Korea, May 15-17, 2014.

Lee, C.S., Wang M.H. and Lan, S.T. (2014c), Adaptive personalized diet linguistic recommendation mechanism based on type-2 fuzzy sets and genetic fuzzy markup language, IEEE Transactions on Fuzzy Systems, (DOI:10.1109/TFUZZ.2014.2379256).

Lee, C.S., Wang, M.H., Wu, M.J., Nakagawa, Y., Tsuji, H., Yamazaki Y. and Hirota K. (2013), Soft-Computing-based emotional expression mechanism for game of Computer Go, Soft Computing, Vol. 17, No. 7, pp. 1263-1282.

Lee, C.S., Wang, M.H., Hagas, H., Chen, Z.W., Lan, S.T., Kuo, S.E., Kuo, H.C. and Cheng, H.H. (2012a), A novel genetic fuzzy markup language and its application to healthy diet assessment, International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems, Vol. 20, No. 2, pp. 247-278.

Lee, C.S., Wang, M.H., Chen, Y.J., Hagras, H., Wu, M.J. and Teytaud, O. (2012b), Genetic fuzzy markup language for game of NoGo, Knowledge-Based Systems, Vol. 34, pp. 64-80.

Melia, M. and Pahl C. (2009), Constraint-based validation of adaptive e-learning courseware, IEEE Transactions on Learning Technologies, Vol. 2, No. 1, pp. 37-49.

Meng D. and Pei, Z. (2012), Extracting linguistic rules from data sets using fuzzy logic and genetic algorithms, Neurocomputing, Vol. 78, No. 1, pp. 48-54.

Mendel J. M. (2001), Uncertain Ruled-Based Fuzzy Logic Systems: Introduction and New Directions, Upper Saddle River, NJ: Prentice Hall, USA.

Munoz-Organero, M., Munoz-Merino, P.J. and Kloos, C.D. (2010), Personalized service-oriented e-learning environments, IEEE Internet Computing, Vol. 14, No. 2, pp. 62-67.

Programme for International Student Assessment (PISA) (2014), Beyond PISA 2015: a longer-term strategy of PISA, website: http://www.oecd.org/pisa/pisaproducts/Longer-term-strategy-of-PISA.pdf.

Ree, M.J. (1997), Estimating item characteristic curves, Applied Psychological Measurement, Vol. 3, No. 3, pp. 371-385.

Sampayo-Vargas, S., Cope, C.J., He, Z. and Byrne, G.J. (2013), The effectiveness of adaptive difficulty adjustments on students' motivation and learning in an educational computer game, Computers & Education, Vol. 69, pp. 452-462.

Thompson N.A. (2009), Ability estimation with item response theory, website: http://www.assess.com/docs/Thompson_(2009)_-_Ability_estimation_with_IRT.pdf.

Wikaisuksakul, S. (2014), A multi-objective genetic algorithm with fuzzy c-means for automatic data clustering, Applied Soft Computing, Vol. 24, pp. 679-691.

Yu, M.N. (2009), Item Response Theory (IRT) and its Application: Psychological Publishing Co., Ltd, Taiwan (in Chinese).

Zarinbal, M., Zarandi, M.H.F. and Turksen, I.B. (2014), Interval type-2 relative entropy fuzzy c-means clustering, Information Sciences, Vol. 272, pp. 49-72.