

Utility Based Agent for Test Paper Generation

¹Memoona Naz, M. Aslam² and Ehtesham-ul-haq Dar³

^{1,2}Department of Computer Science and Engineering, U.E.T., Lahore, Pakistan

³Technical University of Vienna, Institute of Software Technology and Interactive System, Austria

mamoona_mn@yahoo.com

Abstract - Test paper generation for admission in various level of education is one of the most preliminary requirements. Most subsisting systems automate test paper generation by randomly selecting question items from knowledgebase/database. Unlike the existing test paper generators, our utility based agent test paper generator chooses question items in such a way that the difficulty level of each question items takes part in computing exact difficulty level of test paper. So test papers on the same level are different on basis of preferred difficulty level. In design phase, posed utility based model for test paper generation first develops a knowledgebase of question items on which our selection algorithm operates. The utility theory is used in assigning difficulty level of question items. The question items are states which are acting as roads for reaching our goal. Later, a test paper is produced by test paper generator according to difficulty level specified by examiner. Here, selected difficulty level is for the entire paper. Finally, it is implemented, executed, and tested to ensure the feasibility of this approach.

Keywords- Automation, Test Paper, Utility-Based Agent and Knowledgebase

I. INTRODUCTION

With the advent of computer based technology there is evolutionary change in many areas of our professional environments. Most peculiarly e-education and e-learning is highly influenced. There is movement from manual to automated systems for different aspects of education system. These automated systems provide cost saving and time efficient solutions [1]. At every stage/level of education some admission process is required. The key to this admission process is a test paper. For example General Aptitude Tests (GAT) is taken for admission in educational institutes at various level of higher education. These test papers are mostly comprised of short questions, multiple choice questions, etc.

The questions are of logical, mathematical, and English comprehensive type. The patterns for these tests are usually same at different levels except difficulty level changes with the change in level. The most e-systems for generating test papers developed so far composes test papers using question items from already entered in database/knowledgebase. These systems do not take into account difficulty level of question items at the time of composing test papers. The idea of difficulty level is induced in some intelligent systems but it is considered at

assessment time [2]. One multi-agent test paper generator considers difficulty level of question items in test paper generation but it does not provide approximate solution. Moreover, difference in test papers due to difference in difficulty level is not considered at same level [3].

Our proposed system considers difficulty level of questions at the time of test paper generation. The difficulty level assigned at the time of developing knowledgebase takes part in absolute computation of entire difficulty level of test paper.

Utility based agents are used when we are interested in how efficiently our goal is achieved [4]. As discussed, our goal is to generate test with associated difficulty. So, we propose the use of utility based agent for generating test paper.

The remaining material is organized as follows: Section II describes what utility based agents, some application areas and development tools of existing utility based agents and posed UBATG. Section III presents utility based test paper generation algorithm, pseudo code and development of knowledgebase. Section IV demonstrates proposed approach using a case study. Section V narrates related work in context of our UBATG. Finally, conclusions and future work are discussed in Section VI.

II. UTILITY BASED AGENTS

The introduction of idea of utility function in goal based agents provides a new paradigm for agent based technology. This paradigm features autonomous, proactive and reactive behavior in achieving its goal and how well its goal is achieved.

The utility based agents are used where we are not only interested in achieving its goal but also want to determine the measure of how efficiently our goal is achieved. The efficiency criteria can be any parameters we want to achieve in addition to our goal. For example, our goal in this paper is to generate a test paper but its difficulty level specifies what measure of difficulty we want to achieve. Moreover, the usual benefits of agent technology like usability, flexibility, efficiency in terms of cost and time is also proposed by utility based agents.

A. Applications Areas of Utility Based Agents

Till now many utility based agents are developed. Some application areas where utility based agents used are as follows:

- Utility Based Multi-Agent System for Performing Repeated Navigation Tasks [10].
- Utility-based Agents for Hospital Scheduling [11].
- Utility of Mobile Agent Based E-Commerce Applications with Trust Enhanced Security [12].
- Wireless sensor networks, an energy-aware and utility-based BDI agent approach [13].

B. Development Tools for Implementing Utility Based Agents

Since utility based agents are goal based agents with efficiency measure. So, our main concern is on developing tools for achieving the task. Some existing tools that provide reusable components for developing agents are agent Tool Project, ZEUS, Comet Way JAK. However, we adopt object oriented paradigm for implementing our utility based agent.

This implementation is not specific to any development platform like JAVA or .NET. Moreover, the knowledgebase can be in form of file or relational database as we have used object for storing the information about objects in knowledge base. Here, we have used visual C++ for implementing our UAS BASED Test paper Generation and files for the permanent storage of knowledgebase.

III. UAS BASED TEST PAPER GENERATION (UBTPG)

The goal of our proposed utility based agent is to generate a test paper according to the level of difficulty chosen. The test paper is comprised of prescribed number of questions. These questions are acting as states in our domain and are used in computing path for reaching to the destination place.

For UBA to operate, the application primarily requires building a knowledgebase of question items. Then, the examiner develops knowledgebase which enters number of questions appearing in the test paper. The number of questions entered is according to the methodology for knowledgebase. When this task is completed, we have knowledgebase on which our selection algorithm for question items operates. The examiner provides the level of difficulty in a given range. The working UBA starts and the test paper are generated dynamically which confirms the required difficulty level.

A. Knowledgebase Development Using Utility Function

As discussed, the initial requirement of our application is the development of knowledgebase of question items. The required number of question items in the knowledgebase is determined using those entered by examiner. For each test paper the count of the questions can vary. Each question item is assigned an integer value from 0 to 5 at the time of entry. A number in this range represents particular utility value for states considered in domain i.e. question items. Higher the utility value of question, the more it is difficult.

The formula for computing required number of question items in knowledgebase is as under:

$$\text{Total_Questions} = \text{Questions_of_Testpaper} * n$$

Where n is a no. of utility value that can be assigned to any question item where n ranges from 0-5.

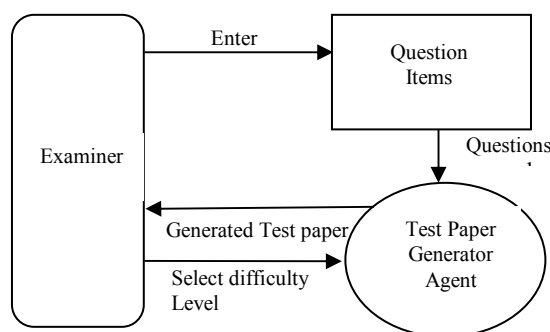


Fig. 1: Working of Utility Based Agent Test Paper Generator

Examiner must enter value for number of questions for any test paper more than 20 (see Figure 1). However, there is no upper limit for any number of questions for test paper. A simple file is used as knowledgebase for question items.

B. Test Paper Generation Algorithm

Once a knowledgebase is developed, the states with associated utility value are available for computing a way to reach our goal. Next step is the choice of overall difficulty level in the given range. The range is from 1 to 100% giving 100 difficulty levels for test paper. The value associated with each level is used in selecting these question items/states for computing overall difficulty level given.

A marked field associated with every question will be set to 1 when it is selected. This ensures no repeating question item will be selected. Every time a question item is selected, the associated utility value is used in computing overall difficulty for test paper. The highest utility value for particular value is calculated as:

$$h_uval = d_level / 20$$

Where 20 is the interval value for difficulty level after that the highest value is changed. Two sets of question items are selected on the basis of utility value. One set of the question items has highest utility value calculated above. The second set of question items has one less than highest utility value calculated. The utility value of question items is used to compute the accumulative difficulty level of test paper. The algorithm is explained using pseudo code as shown in Figure 2.

The approach here not restricts utility value for maximization or minimization. It works only on dynamically given difficulty level.

```

The Pseudo code is as follows:

1. SET dlevel to 0,
   huval to 0,
   i to 0,
   j to 0,
   count to noofquestions
2. READ dlevel //difficult level of
test paper
3. CALCULATE huval = dlevel/20
// highest uvalue for this paper by
usng formula
4. FOR i<20-(dlevel%20)
5. GET Qitem // question items
having highest u-value
6. ENDFOR
7. FOR j<qcount-(20-(dlevel%20)
8. GET Qitem //question items
having one less u-value.
9. ENDFOR

```

Fig. 2: UBTPG Pseudo Code

IV. CASE STUDY

1. In the start of application program prompts for number of questions that arises in test paper as shown in Figure 3. As starting limit for question items that a test paper can contain is 20. So 20 are entered for count of number of questions. This case study uses short answer questions.

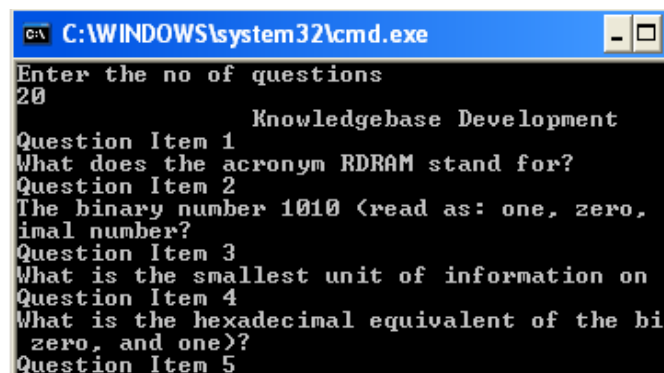
2. As an example, we have entered number of questions that will appear in test paper is 20. The user prompts for 120 questions to be entered according to formula given in knowledgebase development as shown in Figure 3.

3. The program prompts for the percentage of difficulty level for test paper. The user enters value a value 77 as shown in Figure 4.

4. In this case the highest utility value is 4. The 3 questions in this test paper will be value of 3 as given in test generation algorithm. The remaining question will be of utility value 5.

5. Computing these utility values for all questions confirm the overall difficulty level given for test paper.

6. Finally Questions are displayed on screen as shown in Figure 4.

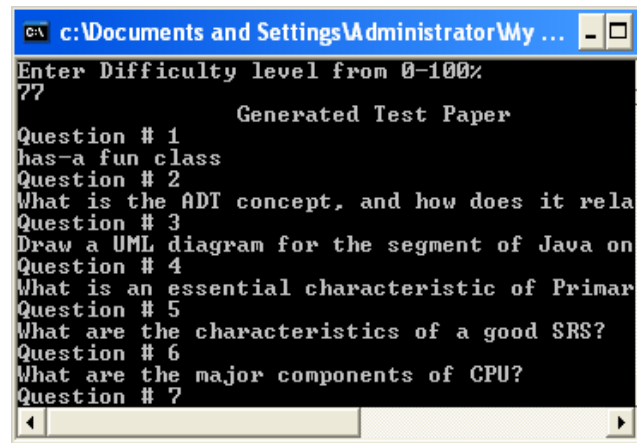


```

C:\WINDOWS\system32\cmd.exe
Enter the no of questions
20
Knowledgebase Development
Question Item 1
What does the acronym RDRAM stand for?
Question Item 2
The binary number 1010 (read as: one, zero,
imal number?
Question Item 3
What is the smallest unit of information on
Question Item 4
What is the hexadecimal equivalent of the bi
zero, and one)?
Question Item 5

```

Fig. 3: Knowledgebase Development



```

c:\Documents and Settings\Administrator\My ...
Enter Difficulty level from 0-100%
77
Generated Test Paper
Question # 1
has-a fun class
Question # 2
What is the ADT concept, and how does it rela
Question # 3
Draw a UML diagram for the segment of Java on
Question # 4
What is an essential characteristic of Primar
Question # 5
What are the characteristics of a good SRS?
Question # 6
What are the major components of CPU?
Question # 7

```

Fig. 4: Generation for Test Paper

V. RELATED WORK

For automation of generation for test papers, many applications were developed. These applications includes from simpler ones to base on artificial intelligence. Some of these simple test paper generators are Paper Builder and QGenie. The Paper Builder provides a test paper generation feature by selecting questions at random from large question bank. QGenie is web based test paper generator. It automates test paper generation by considering parameters like learning objectives, types of questions, competency level and difficulty level. But all these parameters are specified manually by paper setter.

The Intelligent system like IOAS is presented to automate the features like difficulty level, evaluation of students, and improve in learning by considering results. The difficult level is assigned at the time of making question pool but used in assessment only. In this system the preliminary level of students is assessed using some initial test. Then according to calculated level a new question paper is generated using questions from question banks. The students solve the questions, the results are stored and the student levels are updated according to the results. The IOAS is intelligent online system comprising of three components that are Difficulty Assessment Algorithm, Automatic Question Generator, and Intelligent Question Module. The intelligent question module is further divided in two components i.e. the knowledge module and the question module. The question module makes use of difficulty assessment algorithm and automatic question generator.

The level of difficulty in difficulty assessment algorithm is determined by considering number of conditions in questions and number of edges traversed downward and upward in functional graph. In automatic question generator an open queue is used to represent test paper. Each node in queue is question item. The difficulty level of question item is determined by the depth at which it exists. The type and the level of difficulty are taken in account of overall assessment of student's performance in particular test. The knowledge map is used to represent

student's current level of understanding and it also updates level after assessment [2].

A multi-agent approach is used to deal with issues related to test paper generation in distributed environment. The automatic test paper generation using multi-agent approach takes into account the difficulty level of questions and the type. Matrices are used to represent type and difficulty level of question items. In one matrix the percentage of each type of items in question paper is represented and in second matrix the difficult level of each question item of particular type is shown. MASTG calculates approximation of required difficult level and then produces test paper [1]. The multi-agent methodology is more like peer to peer approach in distributed environment

There is also a unique solution which combines genetic algorithm for test paper generation and multi-agent systems for applications in distributed environment. Genetic algorithm is based on survival of the fittest theory and has operations like cross over and mutation. The structure for test paper is defined using a chromosome having fixed length. A test paper is similar to single individual from population. It has some characteristics like type of questions, number of questions belonging to one type and the difficulty level for whole question paper. An integral weight for each type of question item is also assigned as score for that type. A test paper is organized in portions of different type of question items. The score assigned to each type is used in selection and to compute the percentage of portion given. The difficulty level is divided in five portions of difficulty for whole test paper.

Additional features like knowledge points and teacher requirements are also considered. A teacher agent is responsible for providing all parameters. A request for test paper i.e. TPAgent is forwarded to ExamCenterAgent which is real initiator for all agents like CtrlAgent and TPAgent. TPAgent is individual test paper and have information about itself. A TPAgent is produced after considering all constraints. CtrlAgent get information from TPAgent, calculates fitness using fitness function, applies both genetic operators, and again calculates the fitness iteratively for many TPAgents. If TPAgent is not suitable then CtrlAgent kills particular TPAgent. When the whole operation is over, the best solution is sent back. For every request best potential solutions i.e. TPAgents are selected for operations and the others are removed from population on the basis of last evaluations. This technique is based on approximation of fitness value and probabilities of survival i.e. test paper [9]. The multi-agent paradigm reduces issues in traditional client server approach.

Utility based agents provide means to achieve goals with efficiency measures. In some applications this efficiency measure is computation of total expected utility of goal.

An interface agent architecture CIaA deals with the problem of usability in interfaces. The usability is computed using the expected utility. User intentions and path for achieving goals are not pre-planned. The user can take any course of action available for different goals. This behavior

is intentions because of considering both the environmental events and intentions. The main intent is to present this new dynamic model that can calculate expected utility for prediction of user intentions. The CIaA uses Bayesian network based user model as its main component. The random variables in above network for this architecture are goals, actions, and preconditions. A history observation stack is preserved with messages against receiving each event occurring in environment. These events can be precondition/action. Some utility functions are associated with every action variable and are used in computing total expected utility. Whenever, any observation is received from interface it is entered in observable history stack later. This stack is used in updating our Bayesian network. The CIaA agent can act in two ways, offering assistance and autonomously performing actions. The actions are determined on the basis of two things, expected utility and limits. The assistance provided by CIaA is determined by two limits provided by users. The first limit is used in acting autonomously on the user behalf and second in providing help. The autonomous action and assistance against requested message by CIaA is provided on the basis of comparison of expected utility and two limits mentioned above [3].

The agent for dynamic environments where plans for achieving goals are not predictable i.e. mobile agent tracking targets in dynamic environment deals in selection of plan with highest probability of achieving goals along with consideration of efficiency of executing this plan. The efficiency is measured by computing highest expected utility of plan. The total expected utility is computed using the associated utility value of states in world. The problem domain is represented by Bayesian network. The random variable for this network is a plan which takes all possible plans as values for this variable. Initially state with highest utility is considered as final state. Now for this goal a set of candidate plan with highest probability is selected in decreasing order. For every plan expected utility is calculated. Instead of computing actual, expected utility bounds for expected utility are considered. These bounds are refined iteratively [4].

VI. CONCLUSIONS AND FUTURE WORK

This paper gives a novel approach for generating test using utility based agent. This UBATG provides us a great ease in following ways:

- a) The development of knowledge base of question items either using file system or relational database,
- b) Generation of test paper using question items in knowledgebase according to difficulty level selected, and
- c) The final simulation results validated the feasibility of the proposed approach.

By using UBATG algorithm examiner has no need to consider difficulty level of question items at time of test paper generation. The examiner work is reduced to entry of question items. Posed UBATG considers only one part of e-admission system i.e. test paper generation. Furthermore, this single agent system can be extended to multi-agent

system by incorporating exam agents, distribution agents, and assessment agents. The exam agent is responsible for providing different types of question items. The distribution agent concerns with delivering online test papers and submission. The assessment agent can be built for mathematical or derivational type of question items.

REFERENCES

- [1] W. Hairui and W. Hua, "Research and Implementation of Multi-agent Based Test Paper Generation Algorithm," In 2008 International Conference on Computer Science and Software Engineering, IEEE Computer Society, Washington DC, USA, vol. 01, pp 493-496, 2008.
- [2] A. T. Fong, H. H. Siew, P. L. Yee, L. C. Sun, "IOAS: An Intelligent Online Assessment System," WSEAE Transactions on Computers, vol.6, no.3, pp 552-559, 2007.
- [3] Brown, M. Scott and S. Eugene, B. Sheila., "Utility Theory-Based User Models for Intelligent Interface Agents," In 12th Biennial Conf. of the Canadian Society for Computational Studies of Intelligence, 1998.
- [4] J. Kirman, A. Nicholson, M. Lejter, T. Dean, and E. Santos Jr., "Using goals to find plans with high expected utility," In Second European Workshop on Planning, Linkoping, Sweden, pp 158-170, 1993.
- [5] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," San Mateo, CA., 1996.
- [6] T. Li and S. Sambasivam, "Automatically Generating Questions in Multiple Variables for Intelligent Tutoring," Proc ISECON, pp 472 – 478, 2003.
- [7] F. Mills and R. Stuffelbeam, "Introduction to Intelligent Agents," National Science Foundation, 2005.
- [8] B. Coppin. 2004. "Artificial intelligence illuminated," John and Bartlett Publishers, pp 549-551, 2004.
- [9] A. Meng, L. Ye, D. Roy and P. Padilla, "Genetic Algorithm Based Multi-agent System applied to Test Generation," Elsevier Science Ltd, pp 1205-1223, 2007.
- [10] R. Meshulam, A. Felner, and S. Kraus, "Utility Based Multi-Agent System for Performing Repeated Navigation Tasks," In International Conference on Autonomous Agents. ACM publishers, New York, USA, , 2005.
- [11] H. Czap, M. Becker, M. Poppensieker, A. Stotz, "Utility-based Agents for Hospital Scheduling," in First International workshop on multi-agents systems for medicine, computational biology, and bioinformatics. The German National Research Foundation, Germany, 2005.
- [12] C. Lin, V. Varadharajan, Y. Wang, "Utility of Mobile Agent Based E-Commerce Applications with Trust Enhanced Security. Trust," Privacy and Security in Digital Business, Heidelberg, 2005.
- [13] S. Shen, M. Gregory and P. O'Hare "Wireless sensor networks, an energy-aware and utility-based BDI agent approach," In International Journal of Sensor Networks. Inderscience Publishers, Switzerland, , 2007..
- [14] F. logics, "Paper Builder. Foundation for Examination Question Paper generation," 2009. Available from http://www.freshlogics.com/paper_generator.php
- [15] A. land, "Development Tools," 2001. Available from http://www.agentland.com/Download/8How_to_develop_a_n_agent/Development_tools/
- [16] JLLIT, "QGenie. For manual test paper generation," Available from <http://www.qgenie.com/whyqgenie.do>