Potential of knowledge discovery in automated test assembly

Miroslava Ignjatović, Dragan Bojić, Bojan Furlan, and Igor Tartalja

***Abstract*– In the academic environment, acquired student's knowledge is usually assessed through paper-pencil or computer based tests. Testing is usually done several times during the academic year. Assembling a test that estimates students' knowledge requires significant examiner's effort and time invested. While assembling a test, examiner has to be objective, to provide consistent and reliable tests during the whole year. Automating the procedure can facilitate the process to the examiner and improve objectivity, consistency and reliability of student's knowledge assessment. This paper presents the results of development of a test assembly method based on the discovery of test assembly knowledge implicitly embedded in a bank of representative already administered tests. The method uses the nearest neighbour algorithm and it does not depend on a method of creation of existing tests in the test-bank. Results of the conducted experiments are promising and encourage further research in the same direction.**

***Keywords* — automated test assembly, knowledge discovery, knowledge test, nearest neighbour**

I. INTRODUCTION

EDUCATIONAL testing is a common method for students' knowledge assessment in a specific field of interest and it is usually done several times in a year. Manual test assembly is a very complex process [1] which takes time, experts' knowledge and significant efforts. Examiner generally has some global goals and constraints, such as number of items in test and distribution of items by course topics. Based on these criteria, examiner creates new items or reuses old ones to assemble a new test. Difficulty of a new item is typically (in the academic environment, where appropriate control groups of students are practically impossible to organize) estimated based on a subjective item evaluation by an examiner – the item author or his/her colleague. Difficulty of used items is easier to estimate, using statistical analysis based on general-purpose or specific tools such as the one presented in [2]. On the other hand, it is more complicated to model the influence of item exposure to its difficulty that will be exhibited on the next reuse. The examiner has to create several tests during the year, and those tests need to have consistent difficulty. These tests present a kind of parallel test forms, which are sequentially administered. Through automation of the process, test assembly can be facilitated and the quality of a newly created test preserved or improved.

This paper presents early results of the development and evaluation of *testMiner* software tool for automated fixed-length test assembly which makes use of a knowledge discovery technique based on the nearest neighbour algorithm. While other approaches search for a (sub)optimal test, we are satisfied with a test that is similar enough with previously administered tests (by characteristics, not by semantics of the used items). For our approach it is not important how previous tests were assembled – manually or automatically, if they satisfy relevant criteria. These criteria may be empirically determined (a frequent case in academic courses) or theoretically proved (typically for mass stake exams and psychological testing). The proposed method is not sensitive to the method of creation of tests in a test bank. The goal of the presented research was to estimate the potential of knowledge discovery in automated test assembly process while preserving test consistency.

testBase [3] software is used for manipulation of items, topics, tests, and test assembly constraints. The *testMiner* software tool is developed using programming language Java. MS SQL database is used for storing items, tests and criteria.

This paper begins with the problem definition. After introducing the related work, the proposed solution is elaborated. Afterwards, results of the developed tool *testMiner* evaluation are presented. At the end, a conclusion and suggestion for the future research are drawn.

II. PROBLEM STATEMENT

This paper presents our experience in solving the problem of developing an efficient automated test assembly algorithm. The goal of the algorithm is to preserve consistency between newly assembled tests and previously administered ones. Previously administered tests, created manually or automatically, constitute a reference tests set. Automation refers to a procedure of item selection using a set of test assembly criteria (constraints and goals for a solution search), item bank and

Paper received May 7, 2015; accepted September 22, 2015. Date of publication November 15, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jovan Đorđević.

This paper is a revised and expanded version of the paper presented at the 22th Telecommunications Forum TELFOR 2014.

The work presented here was partially supported by the Ministry of Science and Technological Development of the Republic of Serbia (projects TR32039,TR32047 and III 44006).

Miroslava Ignjatović is with ICT college, ZdravkaČelara 16, 11000Beograd, Srbija; e-mail: miroslava.ignjatović@ict.edu.rs; Dragan Bojić, BojanFurlan and IgorTartalja, are with the University of Belgrade – School of Electrical Engineering; e-mail: bojic@etf.rs, bojan.furlan@etf.rs, tartalja@etf.rs.

doi:10.5937/telfor1502108I

reference tests set. The proposed procedure is based on the discovery of knowledge on test assembling, implicitly existing in the reference tests set. Test consistency is achieved using a metrics the calculation of distance between newly assembled test and tests in the reference set, with the goal of having a newly assembled test sufficiently near to the tests from the reference test set. The basic assumption is that tests in the reference set satisfy relevant quality criteria. An immediate goal of this paper is to estimate the potential of knowledge discovery method and the usage of suggested metrics for automated test assembly.

III. Related work

The test assembly problem belongs to the class of combinatorial optimization problems. A solution to the test assembly problem is to find an item set that conforms to a set of constraints and to have defined quality function maximized. Since the number of items in test banks is usually considerably large, finding the best solution would require search of a whole item space, which is an unacceptably long process, because of a NP nature of generation of combinations. For this reason, practical solutions are always suboptimal.

Majority of existing solutions are based on integer linear programming (ILP), where the test assembly problem (objective function and constraints) is represented as a set of linear functions. Special purpose software tools, usually named *solvers*, are designed for solving mixed integer linear programming (MILP) problems and can be also used for test assembly. Some examples are CPLEX [5], Gurobi[6], Xpress [7], which are commercial solvers, and lp_solve[8], GLPK[9], which are free ones. Majority of available solvers are commercial and have excellent performance [10]. Generic problems of these solvers are their cost and usage complexity, which consequently leads to poor usability in academic environment. Free solvers can have problems with modest performances and questionable feasibility due to unsupported solutions for opposed constraints [11].

Besides solvers, which try to search the whole item space in a systematic manner, many local search heuristics, which are satisfied with acceptable suboptimal solutions, such as Hill Climbing[12], Genetic Algorithm[13]-[16], Simulated Annealing[17], Bees Algorithm[18],and Tabu Search [19] are used for test assembly.

IV. Proposed solution

The *testMiner* software tool, proposed in this paper, is purposed for automated test assembly using a knowledge discovery method based on existing items or newly created ones, previously administered tests (which constitute a reference tests set) and criteria that constrain the selection of items for the test (criteria that apply to each particular item and criteria that apply to the whole test). The basic idea of the solution is to find an item set that meets the criteria, and whose distance from the previously held tests is in the range of calculated mutual distance of tests in the reference test set.

A. Item and test attributes

Every test consists of items. Items in the *testMiner* are described with the following attributes:

- Item type: in the used item bank there were two types of items –multiple-choice questions and written (programming) assignments.
- Item topic: every item can belong explicitly or implicitly to one or more topics from a topic-tree. For every course, there is a predefined set of root topics that belongs to it.
- Item difficulty: for items that were already used in some of the previously administered tests, difficulty (0-100%) is calculated from the results achieved on this item, and for new items difficulty is estimated using expert knowledge of the examiner. Difficulty of multiple-choice question is calculated as 100-(percentage of correct answers) and difficulty of assignment as 100-(middle percentage of the fulfilment of evaluated concepts).
- Item discrimination: for new items, item discrimination is set to 0.5 and for already used items it is calculated as described in [20].

Every test or subtest (set of items that are of the same type, and that belong to the same topic) is also described with a set of attributes. The attribute set for describing a test or subtest is the same. Test is described with an attribute set for the whole test and one matrix (number of item types x number of topics for the subject) of attribute sets per each subtest. Every element of the matrix is an attribute set describing the subtest that consists of items of one type and the same topics. The following attributes of a test/subtest are observed:

- Number of items: number of items in the test, or subtest
- Test type: identifier which implies test type (for example: mid-term 1, mid-term 2, final exam).
- Minimum difficulty: minimum difficulty of an item in the test or subtest
- Maximum difficulty: maximum difficulty of an item in the test or subtest
- Average difficulty: the mean difficulty of items in the test or subtest.

B. Filtering items and candidate-tests

Acceptable ranges of values for item and test attributes are determined on the basis of the reference test set. The reference test set consists of all previously administered tests for the given course and test type. The calculated limit values of attributes are used as criteria to filter items and tests in order to reduce the search space. There are two types of criteria for filtering:

1. Individual eliminatory criteria (IE). IE criteria serve to filter items belonging to the topic: topic of an item must belong to the course. Items with an attribute value out of the range are eliminated from further consideration. Filtering is done at the beginning, before generating the first candidate-test.
2. Group eliminatory criteria (GE). GE criteria refer to the test as a whole. They prevent the occurrence of a

combination of items that do not meet them. The values of the following test or subtest attributes should be in the range of values determined as described above:

a. The test difficulty;
b. The number of items per topic;
c. The difficulty of each subtest;
d. Test information function (TIF) defined in Item response theory (IRT) [21].

Also a GE criterion is used to constrain the repetition of items already used in previously administered tests: the number of items of the same type that have appeared together on a test from the reference test set should not exceed half of the total number of items of this type in the new test.

If a candidate-test does not meet the GE criteria, it is eliminated from further consideration.

C. The test assembly procedure

The test is assembled as follows:

1. At the beginning, a target course and test type is defined, as well as the number of items per item type (for example, the final exam for the course Programming 1 with 5 questions and 2 programming assignments). Also, the target number of satisfactory candidate-tests as the criterion for the termination of the procedure is defined. A satisfactory candidate-test is the candidate which doesn't violate eliminatory criteria and whose distance from the reference test set is in the range $[a-\sigma,a+\sigma]$, where a denotes the average distance between tests within the reference test set, and σ denotes the standard deviation of distance.
2. Based on the given course and test type, the reference test set is formed. Then, the acceptable range of test attributes' values (which are used as parameters for GE criteria) is calculated, as well as the average distance between the reference tests, along with the standard deviation of distance.
3. Items are filtered based on IE criteria. It is checked only if item topic belongs to the topics of the course for which the test is assembled. The item difficulty is not checked, in order to allow easier and more difficult items than those in reference tests, to appear in a newly assembled test.
4. Items that pass the IE criteria are not ranked, but pseudo-randomly selected and placed in an array of a list of items; items in one list belong to one item type.
5. A candidate-test is assembled. From the array of the list of items, items for the new test are selected using the procedure which generates the next combination. The procedure uses the principle to include items with a greater index as late as possible in the combination. For example, if the array has two elements, the first with questions V1={P0,P1,P2,P3,P4} and the second with programming assignmentsV2={Z0,Z1} and if the new test needs to have two questions and one programming assignment, the procedure would generate next candidate-tests as follows: P0P1Z0, P0P1Z1, P0P2Z0, P0P2Z1, P1P2Z0, P1P2Z1, P0P3Z0, P0P3Z1, P1P3Z0, P1P3Z1, P2P3Z0, P2P3Z1, P0P4Z0, P0P4Z1, P1P4Z0, P1P4Z1, P2P4Z0, P2P4Z1, P3P4Z0, P3P4Z1.
6. Each candidate-test is evaluated according to the GE criteria. If a candidate-test does not meet the GE criteria, step 5 is repeated and the next candidate-test is generated.
7. If the candidate-test meets the GE criteria, the average distance between the candidate-test and tests from the reference set is calculated. If this distance is in the defined range (parameters determined in step 2), the candidate-test becomes a satisfactory-test. It is stored as the selected candidate if its distance is lower from the distance of the previously selected candidate.
8. If a satisfactory test is not found in a given time period (default 5s), the procedure is repeated from step 5 (random-restart). Items are again pseudo-randomly placed, and the combinations begin to be generated from a new starting one. The new combinations are different from the previous, due to the different order of items in the list.
9. The process ends if the given number of satisfactory tests is assembled. The result of the procedure is the latest stored selected candidate-test.

D. Measuring the distance between two tests

The distance between two tests is measured using test attributes' values. Each test is described with the attribute set and the matrix of subtests attribute sets. Distance (d) between the two tests is calculated according to equation 1:

$$\mathrm{d}^{\mathrm{Total}}(\mathrm{x}^*,\mathrm{y}^*) = \frac{1}{2}\mathrm{d}^{\mathrm{test}} + \frac{1}{2*\mathrm{k}*\mathrm{m}}\sum_{\mathrm{j}=1}^{\mathrm{k}*\mathrm{m}}\mathrm{d}_{\mathrm{j}}^{\mathrm{subtest}} \quad (1)$$

where k is the number of item types and m is the number of course topics. Values d^{test} and $d^{subtest}$ are calculated according to equation 2 for Euclidian distance:

$$d(x^*,y^*) = \sqrt{\sum_{i=1}^{n} w_i (x_i^* - y_i^*)^2} \quad (2)$$

where x^* and y^* are vectors of normalized attribute values of the two tests whose distance is calculated, and w_i is a weight factor of an i-th attribute, while $\sum_{\mathrm{i}=1}^{\mathrm{n}}\mathrm{w}_{\mathrm{i}} = 1$. Default values of weight factors for attributes: minimum difficulty, maximum difficulty, the average difficulty, and number of items are 0.2, 0.2, 0.2,and 0.4,respectively.

In order to ensure equal influence of all test attributes (excluding from the consideration the weights that are explicitly defined) on determination of the distance of the candidate-test from the reference set, it is necessary to normalize attributes, i.e., their mapping to the range [0,1]. We used MIN-MAX normalization, and the normalized attribute value is obtained by equation 3:

$$x^* = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (3)$$

Boundary attribute values are taken as the min and max values of attributes.

V. Results

Our experimental database contained items and tests (exams and mid-terms) from courses Programming 1 and Programming 2, which were administered in the period from February 2004 to October 2012 at the School of Electrical Engineering, University of Belgrade. The database had 150 tests and 1075 items. Tests were

administered to 7500 students. There were 34 topics in the tests.

As an example, the results of test assembly for the course Programming1 are presented. The assembled test contains 6 multi-choice questions and 2 programming assignments. There are 25 tests in the appropriate reference test set with the same number of questions and assignments. *testMiner* was configured to search for 100 satisfactory test candidates.

After initial filtering of the items, based on the course topics, there were 437 questions and 118 assignments left. For search of the whole search space, with the goal of finding a combination with a minimum distance we would have to calculate the distance the following number of times:

$$\binom{437}{6} * \binom{118}{2} = 9{,}34 * 10^{12} * 6903 = 6.45 * 10^{16}$$

Because of the exponential nature of the problem complexity, it was impossible to search the whole item space in real time so combinations were generated until a defined number of satisfactory candidate-tests were found, with the help of *random-restart* technique.

Fig.1 presents average distances of 25 tests from the reference test set as well as the average distances between 25 tests assembled using the proposed method and reference test set. The average distance for the reference test set is 0.155 with a standard deviation of 0.028. The average distance for the assembled test set is 0.157with a standard deviation of 0.0098.

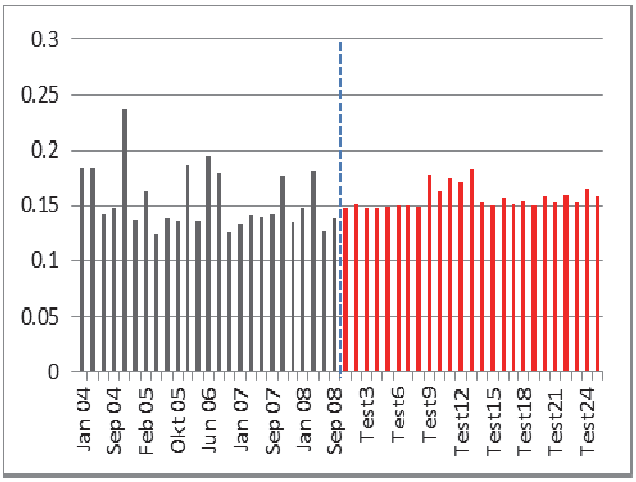

Fig.1. Average distance between tests in the reference test set (left) and average distance between automatically assembled tests and reference test set (right).

Table 1 presents the results of F-test, which is used to determine the statistical significance of the standard deviation of distances for candidate tests that have met only GE criteria, and for a set of satisfactory tests which meet GE criteria and have an acceptable distance from the tests in the reference test set, compared to the standard deviation of test in a reference set.

TABLE 1: RESULTS OF ONE-WAY F-TEST.

	Tests in the reference set	***Candidate-tests that meet GE criteria***	***Satisfactory tests***
Average distance	0.155	0.174	0.157
Variance	$7.9 \cdot 10^{-4}$	$6.0 \cdot 10^{-4}$	$9.6 \cdot 10^{-5}$
Number of tests	25	40	25
F		1.312	8.22
F_{crit}		1.8	1.98

The result of F-test shows that the difference of standard deviation of the distance in a reference test set from the standard deviation of the distance of candidate-tests that meet only GE criteria is not statistically significant, because $F < F_{crit}$. However, the difference of the standard deviation of the distance in a reference test set from the standard deviation of the distance of satisfactory tests is statistically significant, because $F > F_{crit}$. This observation justifies the application of the method for automated test assembly that is proposed in this paper.

For determining the quality of a newly assembled test, a metric of IRT testing theory was used. In IRT, the value of TIF in several ability points is used as a quality measure. First, for a reference test set, average TIF for every ability point was calculated and it was used as GE criteria. Test with a value of TIF lower than the average TIF from a reference test set in the same ability point was discarded. In *Fig.2.* TIF values for 10 newly assembled tests are presented. In the same figure, with dashed lines are presented minimum, average and maximum TIF functions for a reference test set. We can conclude that automating the process as described in this paper, we assembled tests with considerably consistent and undoubtedly improved quality.

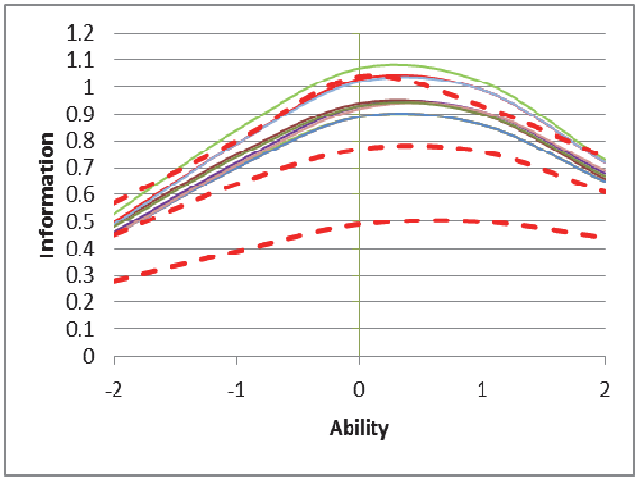

Fig. 2. Test Information Function for 10 newly assembled tests.

CPU time needed for solving this model using PC computer with I5 CPU 1.7GHz and 6GB RAM was between 1s and 20s. Performance depends on the initial random placement of items in an array of items, but the reported values indicate an acceptable time for the automatic assembly of the test, even in the worst case.

VI. Conclusion

The presented process of automated test assembly considerably simplifies the knowledge test preparation process and prevents examiner's subjectivity when selecting items from a bank of test items. The method substantially differs from the existing test assembly methods. It discovers knowledge embedded in the set of previously assembled tests (manually or by any other automated tool) and assembles a new test in a way that established similarity criteria are fulfilled. Our experiments have shown that the technique has the potential for practical use. Assembled test quality and consistency, as well as the tool performances achieved, encourage the usage of the tool. Also, this fact opens up a new avenue for research directed to the implementation of better search heuristics.

Since there were not many reused items in the used item bank, it was not possible to examine the influence of item attributes relating to the item exposure, such as the number of occurrences and the time since the last occurrence of an item on a test. Although there are no reasons to believe that the conclusions would change if a test bank that contains reused items were used in experiments, it should be particularly investigated in the next development phase.

Acknowledgements

Marko Mišić and Igor Anđelković provided a real bank of reference tests set for this research.

References

[1] W. Linden,"Linear Models for Optimal Test Design,"*1st ed. New York, NY: Springer Science+Business Media, Inc.*, 2005.

[2] M. Mišić, M.Lazić, J.Protić,"A software tool that helps teachers in handling, processing and understanding the results of massive exams,"*Proceedings of the Fifth Balkan Conference in Informatics. ACM*, 2012.

[3] J. Protić, D. Bojić, I. Tartalja,"test: Tools for evaluation of students' tests - a development experience,*"Proceedings 31st ASEE/IEEE Frontiers in Education Conference*, Reno, NV, USA, pp.F3A-6–F3A-12*,* 2001.

[4] A. Bošnjaković, J. Protić, I. Tartalja,"Development of a software system for automated test assembly and scoring,"*Proceedings of Int'l Conf. on Education, Research and Innovation (ICERI* 2010*)*, Madrid, Spain, pp. 6012-6016, 2010.

[5] IBMILOGCPLEX Optimization Studio, Available: http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud, last access 05.05.2015.

[6] Gurobi Optimization Inc. Gurobi Optimizer, 2012, Available: http://www.gurobi.com/, last access 05.05.2015.

[7] Fico. Xpress Optimization Suite, 2012,Available: http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx,last access 05.05.2015

[8] lp_solve 5.5, Available: http://lpsolve.sourceforge.net/, last access 05.05.2015.

[9] A.Makhorin. The GNU Linear Programming Kit (GLPK),*GNU Software Foundation*, 2000, Available: http://www.gnu.org/software/glpk/glpk.html, last access 05.05.2015.

[10] B. Meindl,M.Templ. "Analysis of commercial and free and open source solvers for linear optimization problems,"*Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 2012.

[11] H. Huitzing,, B. Veldkamp, A.Verschoor. "Infeasibility in automated test assembly models: A comparison study of different methods,"*Journal of Educational Measurement 42.3*, pp. 223-243, 2005.

[12] D. Bojić, P. Cerović, J. Protić, I. Tartalja, "testGEN: A Software Tool for Semiautomatic Quiz Generation,*"Proceedings 3rdSymp. YUINFO*, Brezovica, pp. 687-691, 1997. (in Serbian)

[13] A.Verschoor, "Genetic Algorithm for Automated Test Assembly,*" Ph.D. dissertationUniversiteitTwente*, 2007.

[14] K. Sun, Y. Chen, S. Tsai,C. Cheng, "Creating IRT-based parallel test forms using the genetic algorithm method,"*Applied measurement in education*, 21(2), pp.141-161, 2008.

[15] M. Yildirim, "A genetic algorithm for generating test from a question bank," Computer *Applications in Engineering Education*, 18(2), pp.298-305, 2010.

[16] M. Yildirim,"Heuristic optimization methods for generating test from a question bank,"*MICAI2007: Advances in Artificial Intelligence. Springer Berlin Heidelberg*,pp.1218-1229,2007.

[17] H. Jeng, S. Shih, "A Comparison of Pair-Wise and GroupSelections of Items Using Simulated Annealing in AutomatedConstruction of Parallel Tests," *Psychological Testing, vol. 44, no. 2,* pp. 195-210, 1997.

[18] P. Songmuang, M. Ueno,"Bees algorithm for construction of multiple test forms in e-testing,"*IEEE Transactions on Learning Technologies* 3, pp. 209-221, 2011.

[19] G.J. Hwang, P.Y. Yin, S.H. Yeh, "A Tabu Search Approach to Generating Test Sheets for Multiple Assessment Criteria," *IEEE Trans. Education, vol. 49, no. 1,* pp. 88-97, 2006.

[20] T.Winters, T. Payne,"What do students know?:an outcomes-based assessment system,"*Proceedings of the first international workshop on Computing education research*,ACM, 2005.

[21] F.M. Lord, "Applications of item response theory to practical testing problems,"*Routledge*, 1980.