ISSN 2395-8618

A Method Based on Genetic Algorithms for Generating Assessment Tests Used for Learning

Doru Popescu Anastasiu, Nicolae Bold, and Daniel Nijloveanu

***Abstract*—Tests are used in a variety of contexts in the activity of everyday and everywhere learning. They are a specific method in the process of assessment (evaluation), which is an important part of the educational activity. Setting an optimized sequence of tests (SOT) originating from a group of tests which have the same subject, with certain restrictions corresponding to a certain wish of the evaluator can be a slowly time-consuming task, because the restriction can be various and the number of tests can be high. In this matter, this paper presents a method of generating optimized sequences of tests within a battery of tests using a genetic algorithm. We associate a number of representative keywords with a test. The user expresses the restriction by setting up a number of keywords which approximate best the subject wanted to be tested. The genetic algorithm helps in finding the optimized solutions and uses a less amount of hardware resources.**

***Index Terms*—Test, genetic algorithm, keyword, sequence, generation.**

I. INTRODUCTION

THE process of learning is very complex and has three major components: teaching, learning and evaluation. This paper focuses on the third component and, in particular, on tests and their usage based on a customization given by the user, taking into account keywords, which are part of natural lexicon. We can specify that the solution proposed in the paper is immediate and needful.

Firstly, the emergence and fast development of devices creates a learning current which have some particularities (fast learning, huge amounts of data obtained relatively fast, knowledge based on competences, less than on accumulation of information etc.). This solution adapts on this intrusion of technology in learning.

Moreover, the time and energy consumed for the manual trial of the tests is obvious. Finding a solution which decreases this time is a need in a climate where every minute counts. For example, for a usual exam, a teacher creates tests with 10 questions that form a battery of tests. These tests have various subjects. In the process of creation of tests, the teacher applies "labels" to each test. For a manual search of tests, we assume that the verification for one test consumes an average of 1,5 minutes. For a battery of 100 tests, the verification consumes a total of 150 minutes. Using this application, the generation and the identification of tests within a well-organized battery consumes maximum 10 minutes. Furthermore, the teacher must make an additional search for different labels, while the generation is made faster.

From the point of view for the student, the solution presented in this paper adapts to the adaptive learning style, presented in detail in section 2. In short, the student can organize the self-assessment process (e.g., learning for an exam) on subjects. Furthermore, if a battery of tests contains tests with various subjects and the student wishes to prepare only for certain subjects in a limited time, the problem escalates. For example, in a battery of 1000 tests regarding programming languages, containing tests about C++, Java, PHP and JavaScript, the student must verify knowledge about Java. The solution can be applied for solving this problem.

Regarding types of tests, they can be various (simple questions tests, multiple-choice tests etc.). But, generally, as human beings, we learn things every moment. This learning process, also called self-learning, includes a constant permanent self-evaluation, even if it is formal, informal or our teacher is ourselves. We all participated at least at one exam or test – this can be called a formal evaluation. Our knowledge is tested by creating things and dealing with situations in real life and this could be considered a less formal examination. The perception is different for students and tutors and, in this matter, a study in the paper [1] shows these differences in perception.

The evaluation is made through several methods, some traditional, some novel (a presentation of new methods in assessment can be found in the book [2] and the perspective of the students on the new methods can be found in [3]). One of the traditional ones is the test (even if it is a multiple-choice test, a question test or problem solve test). Tests can be categorized in a formal register, when the mark obtained by the learner is important for a legal purpose, but also in a less informal register, in case of self-learning and self-evaluation. The purpose is to generate optimized test sequences which

Manuscript received on February 24, 2016, accepted for publication on June 16, 2016, published on October 30, 2016.

Doru Popescu Anastasiu is with the University of Pitesti, Faculty of Mathematics and Computer Science, Romania (e-mail: dopopan@gmail.com).

Nicolae Bold is with the University of Agronomic Sciences and Veterinary Medicine Bucharest, Faculty of Management, Economic Engineering in Agriculture and Rural Development, Slatina Branch, Romania (e-mail: bold_nicolae@yahoo.com).

Daniel Nijloveanu is with the University of Agronomic Sciences and Veterinary Medicine Bucharest, Faculty of Management, Economic Engineering in Agriculture and Rural Development, Slatina Branch, Romania (e-mail: nijloveanu_daniel@yahoo.com).

ISSN 2395-8618

contain the maximum number of keywords from the keywords set by the user.

The learners deal with a high number of tests in their learning process. In some cases, the tests are grouped in batteries or clusters of tests. These tests are not related at all, as in the problem studied in paper [4], where an arborescent structure existed. The particular grouping of tests in clusters makes difficult the process of finding tests with a certain subject. Thus, this paper presents a solution of a fast finding of wanted tests with the desired subject. This is made by previously assigning keywords to tests, then giving some keywords as input data and finding the ones whose keywords match with the given keywords. Thus, the problem is solved using lexical resources. Moreover, the usage of keywords encapsulates the concept of summarization. The results are given in the form of optimized sequences of numbers which codify the tests.

Regarding the problem of summarization and lexical resources, we can say that keywords represent the words that are essential for defining a test. They summarize best what a test contains, turning into lexical resources which are key elements in the solution of the problem.

The optimality refers to the finding of the maximum number of tests that can be selected after the required conditions are respected. Thus, the fitness is a maximum function and the problem can be classified as an optimization problem.

Even if the issue does not appear to be immediate and needful, the trial of tests consumes time and energy. In this matter, an algorithmically solution of this problem would bring a plus of efficiency in the process of evaluation, as shown in the previous paragraphs. The genetic method for generating the sequences was chosen for its performance in the usage of fewer hardware resources and for its variability of output solutions.

Section 2 will contain a short description of testing method and some measurable data to show its efficiency in the evaluation process. In Section 3, we will present some notions regarding genetic algorithms and their role in solving problems. The algorithm description takes place in Section 4 and Section 5 will describe some results regarding the algorithm efficiency and the productivity of the algorithm in solving the problem. Section 6 will draw the lines of this paper and will present the future work which will be made in this matter.

II. On the Impact of Tests, Evaluation and Information and Communication Technologies (ICT) on Education

The main idea of this paper related to evaluation process, in particular, and education, in general, can be viewed from various angles and has more perspectives.

One of them is the personalization of the subjects desired to be learned, either by a teacher or by a student. Regarding the latter category, a specific type of learning style has been implemented in several universities or faculties worldwide, where the student chooses the subjects he wishes to use in the future and the education is made using computers as teaching devices. Roughly, this process can be called adaptive learning style. In a similar way, choosing a sequence which suits best to specific needs foe evaluation will maximize the effectiveness in learning. In a study made in the paper [5] a comparison between an adaptive learning style and a non-adaptive one is realized. This study, based on measuring some key characteristics that influence learning, revealed that a high percent of students (90.63%) think that learning styles are important for learning. Another conclusion of the study is focusing on the fact that students would prefer to have recommended paths for learning, but this should be chosen by the student. The fact of self-choices has several aspects and part of them have not positive effects over the learning status of the students. Thus, some major risks in this case are:

- faulty choices of what is needed for learning due to the immaturity of children or depending on the pupil/student personality;
- lack of human communication, which is partly substituted with modern communication technologies, and its further implications.

Another perspective of using Information and Communication Technologies (ICT) in learning and evaluation is the perception of the student on the teaching style, because of the modern characteristic of using technology [6]. Because they are more familiar with the technology, students perceive this step to modern techniques of teaching as receptivity to innovation from the teacher. From a statistical point of view, in the same study, made on a total of 226 students, it was shown the attitude of students towards the school activity has also shown to be improved in the studied group.

A very interesting perspective is the creation of an open learning environment [7], which appears in case of using technology in education. The development of technology and the equipment, which has more and more influence on the society, in general (Internet, educational platforms and software on one hand and gadgets such as tablets, smart phones, laptops, projectors on the other hand), brings into attention notions regarding a specific new environment, defined in this paper as a combination between social technological and pedagogical factors which influence each other and which influence educations.

Examples of open-learning environments are the e-learning platforms. Studies regarding the inclusion of ICT in education have led to the apparition of e-learning platforms. The usage of Internet in the process of education is more and more visible today. An example of a study and of a model for an e-learning platform, with its threats is detailed in papers [8] and [9].

ISSN 2395-8618

Regarding the particular process of assessment, in the context of the combination between the three processes of education, numerous results of studies and data from literature show the fact that ICT has proven its benefits to evaluation in the matter of online and technology-based assessments versus the traditional methods of student evaluation, as proved in [10], [11] or [12]. Despite these results, there is not known the effect on a long-term period, so we should monitor the effects on a longer period than the time accorded for learning for an exam, possibly showing their efficiency in more practical situations.

The perspectives on ICT influence on the educational process are numerous and have both positive and negative implications on the personality of the student. As the authors in the paper [13] show, there are many questions to be answered in the matter of efficiency of ICT based methods in teaching, learning and evaluation. Thus, we must find answers to questions such as "how can the problem of communication be solved?", "how can practical abilities (e.g., crafting) can be developed?", "how can we measure the efficiency and the added value of educational methods based explicitly on ICT?" or "how can we improve the security of assessment structures?".

III. GENETIC ALGORITHMS

Problems which appear in practice can be solved using different methods, algorithms and structures. Their variety is wide, starting with trees and graphs, continuing with backtracking and greedy algorithms and finishing with random or genetic methods of solving problems. Among them, genetic algorithms are used for problems which solve the optimization aspects. They are inspired by genetics and use notions such as chromosomes, genes, mutation and crossover. Next, we will present shortly some problems that can be solved using genetic algorithm, as well as some key characteristics of them.

Firstly, a genetic algorithm uses a lower amount of hardware resources (the runtime is lower). This is a major point in using genetic algorithms, because of its methods, being preferred to other optimization or heuristic algorithms (such as backtracking) in some cases (for a larger number of solutions, in case of large amounts of input data etc.)

Another characteristic of a genetic algorithm is that the problem solved through this method has to be or to be transformed into an optimization problem [14]. This means that genetic algorithms found the most optimized solution in case of minimum and maximum issues.

An eventual drawback could be the fact that genetic algorithms do not find the most accurate solution, but at least they generate solutions that can be found in its proximity. Genetic algorithms are used mostly in cases in which input data is in large quantities, which is the case of the most practical problems needed to be solved.

One of the areas in which genetic algorithm is used is designing constructions. In the paper [15] there is presented a solution for designing the optimized thermal and lightning conditions, construction materials etc. within a building, using genetic algorithms. GA are also used in issues related to domains such as mathematics, statistics, physics, engineering, transportation, pollution cases [16], chemistry [17], agriculture [18], web programming [19], web applications [20] and even fashion issues [21]. This is a very short list of the applications of genetic algorithms in specific domains.

Even with the drawbacks of this method (the found solutions are optimized, but not optimal), we chose genetic algorithms because they offer a variety of solutions with given restrictions, which is the case of our problem. The large number of tests within a battery and the correspondence of genetic structures with the ones of the studied problem are other reasons for choosing genetic algorithms, besides the ones presented in the introduction.

IV. ALGORITHM

As we said before, the algorithm uses genetic notions inspired by biology and in this way it generates the sequences we want to obtain. In another paper [1], the case in which the tests have a tree-designed relationship was studied. Here, we will show the algorithm when tests are not level-related.

The tests will be codified by numbers from 1 to n. The optimized sequence of tests (a chromosome) is an arrangement with k elements, m being given by the user (representing the number of tests within the test battery) of the set {1,2,…,n} and a test (a gene) within an optimized sequence of tests is represented by a number from the set {1,2,…,n}. The fitness for a sequence is represented by the maximum number of keywords within the sequence (the m-arrangement) which correspond with the keywords set by the user. The chromosomes will be ordered by the value of the fitness function. In Figure 1, an example of the structure of a chromosome with 6 genes which will be output as solution is presented.

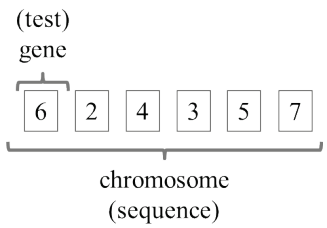

Fig. 1. An example of the general form of an optimized sequence of tests (chromosome) with 6 tests (genes)

Firstly, an array of arrays is initialized with 0 or 1 value (false or true). The purpose of this array is to verify if a keyword which characterizes a test is part of the keywords set

ISSN 2395-8618

by the user. Then, a number from 1 to n, representing a test, is randomly generated and verified if one of its keywords is part of the list of keywords set by the user. After this verification, when the optimized sequence of tests is generated completely, the fitness of this sequence is calculated. The optimized sequences of tests are ordered by the value of the fitness. During the next step, there are made operations such as mutation or crossover between chromosomes (sequences). Then, the resulted sequences are ordered again by the value of the fitness.

The variables used in the algorithm that we will need is presented next:

n: the number of tests within the battery;
m: the number of tests needed in the optimized sequence of tests;
no_generations: the number of generations used to generate the optimized sequences of tests
no_words: the number of keywords set by the user;
no_cuvT: the number of keywords of each test.

The structures (arrays) that will be needed within the algorithm are:

TG[no_words]: the array contains the keywords set by the user;
pop[no_generations][m]: the solution array;
T[n][no_words]: an array of arrays, which has the meaning:

$$T[i][k]=\begin{cases}1, \text{ if the } k^{th}\text{keyword from the } i^{th}\text{test}\\ \quad \text{is found among TG array}\\ 0, \text{otherwise}\end{cases};$$

$$i=\overline{1,n};\ k=\overline{1,no\_cuvT}$$

After presenting the input data, we can say that the fitness has the next form:

$$maxf = \sum_{\substack{1\le j\le m\\ 1\le k\le nr_{cuv}}} T\big[pop[i][j]\big][k];$$

$$i = \overline{1, no\_generations}$$

The input data will be formed from n, m, no_generations, no_words, no_cuvT and the keywords for each test. For avoiding comparing each time the keywords, the array T is built, so there are stored only the keywords which match with the ones from the array TG and the tests containing them. The output data will contain the first k solutions (optimized sequences of tests) from the array pop, where k is a value set by the user, and the number of keywords matching with the ones from TG for each sequence.

After we presented the input and output data needed for our algorithm, we shall present the steps of the algorithm.

Step 1. Input data (mentioned before) is read.

Step 2. The array T is initialized with 0 or 1 (false or true) values, according to the definition presented before.

Step 3. The chromosomes (optimized sequences of tests) are randomly generated, gene by gene. This will be the initial population.

Step 4. The fitness function is calculated for each chromosome. The fitness function is stored in the (m+1)th element of the solution array (pop).

$$pop[m+1] = \sum_{\substack{1\le j\le m\\ 1\le k\le nr\_cuv}} T\big[pop[i][j]\big][k];$$

$$i = \overline{1, no\_generations}$$

Step 5. Operations (mutation and crossover) are applied on the generated chromosomes. The fitness function is calculated for each chromosome in this step too. The fitness function is also stored in the (m+1)-th element of the solution array. Figure 2 presents an example of mutation within a chromosome.

Fig. 2. Mutation within a sequence (chromosome)

Figure 3 presents an instance of the crossover operation with one point between two chromosomes.

Fig. 3. Crossover with one point between two optimized sequences of tests (chromosomes)

Step 6. The chromosomes are ordered by the value of the fitness. At this step, method of order can be used. Steps 5 and 6 are repeated for no_generations times.

Step 7. The first desired solutions are output.

V. Results and Discussion

To show the efficiency in solving a practical problem, we shall take a short example. The battery of test will have as main subject programming languages. Tests have subjects represented by keywords such as software, editing, office, complier, pascal, Java, similarity, syntax, logical, expression, operation, type, protocol, Internet etc., in subdomains such as software in general, programming languages, syntax of languages, memory usage, programming methods, Internet, web programming or database concepts.

ISSN 2395-8618

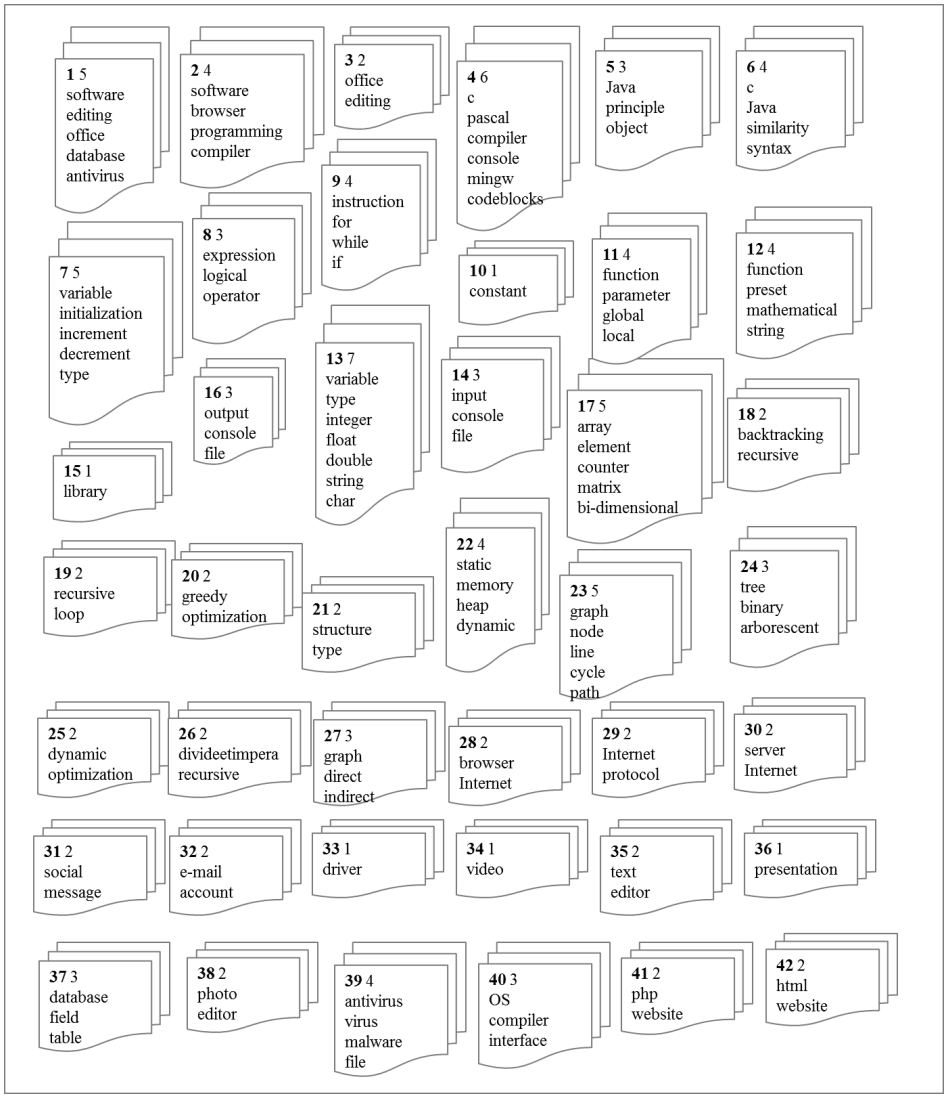

Fig. 4. Battery of 42 tests for our example

For our example, the variables have the next values: n=42, m=15, no_generation=600, no_words=15. The keywords set by the user are:

TG=(software, programming, c, Java, method, structure, variable, backtracking, tree, binary, compiler, instruction, recursive, console, Internet)

The battery of tests contains 42 tests and the keywords that characterize each test are presented in Figure 4. The number in bold characters is the integer assign for the test and the second number represents the number of keywords representative for each test.

The results are divided in two components:

– the output optimized sequences of tests;
– the runtime of the algorithm.

The output in our case is presented in Table II. In case of runtime, we present its dependence on parameters such as the number of tests (n), the number of generations and the number of tests in the optimized sequence of tests (m).

The values were obtained with a code written using the Java programming language, within a Java environment (NetBeans IDE 8.0.2).

The array T stores for each test the keywords that match the keywords from TG. In Table I and II the array T for our example is presented.

The number of keywords from TG which are found in the 42 tests is 22. The first 10 optimized sequences of tests resulted after the program runs are presented in Table III.

The runtime after running the program is 3.877521196 seconds, which shows that the program is efficient regarding the usage of resources. For supporting this affirmation, we present a graph which shows the runtime for different values of n. The values are resulted for m=15 and for no_generations (number of generations) equal to 700. Individual values are an average of 5 values. In the graphs there are made some

ISSN 2395-8618

TABLE I
The array T after running the algorithm

No.	TG 1 2 3 4 5 6 7 8 9 0 11 12 13 14 15
T1	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 TG1 = software in test 1
T2	1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 TG1,2,11 = software, programming, compiler in test 2
T3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 3
T4	0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 TG3,11,14 = c, compiler, console in test 4
T5	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 TG4 = Java in test 5
T6	0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 TG3,4 = c, Java in test 6
T7	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 TG7 = variable in test 7
T8	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 8
T9	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 TG12 = instruction in test 9
T10	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 10
T11	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 11
T12	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 12
T13	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 TG7 = variable in test 13
T14	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 TG14 = console in test 14

No.	TG 1 2 3 4 5 6 7 8 9 0 11 12 13 14 15
T15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 15
T16	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 TG14 = console in test 16
T17	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 17
T18	0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 TG8,13 = backtracking, recursive in test 18
T19	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 TG13 = recursive in test 19
T20	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 20
T21	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 TG6 = structure in test 21
T22	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 22
T23	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 23
T24	0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 TG9,10 = tree, binary in test 24
T25	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 25
T26	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 TG13 = recursive in test 26
T27	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 None of TG elements in test 27
T28	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 TG15 = Internet in test 28

calculus to show the models of the functions. Figure 5 presents the model of the algorithm runtime for different values of n.

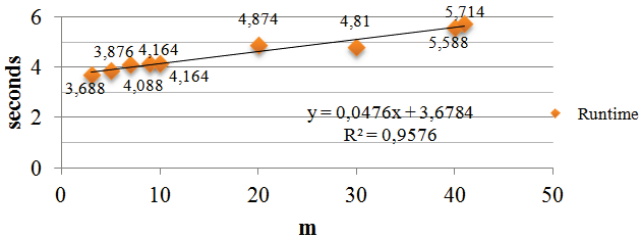

Fig. 5. Runtime depending on the value of m (n=42, no_generations=700)

Figure 6 presents the model of the algorithm runtime for different values of *number of generations*.

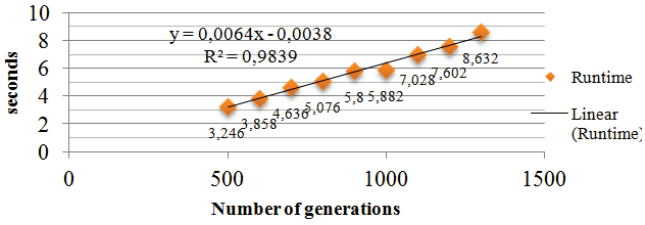

Fig. 6. Runtime depending on the value of no_generations (n=42, m=15)

Figure 7 presents for the model of the algorithm runtime for different values of *m*.

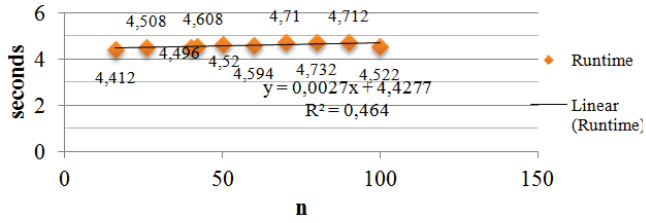

Fig. 7. Runtime depending on the value of n (m=15, no_generations=700)

It can be seen that the models have quite a major significance, depending on the coefficient of correlation R2, especially for models from Figures 6 and 7. All three models are based on a linear regression.

As we can see, the number of tests within the battery does not influence very much the runtime of the algorithm. The number of genes in the chromosome (m) influences in a minor way the runtime, the biggest increase being shown in case in which we want more accurate solutions (in this case, the number of generations increases). Despite this increase, the difference between the first measured element and the latter is 5.836 seconds (from 500 to 1300 generations). This is not an extremely significant difference, given the fact that the number of generations almost triples.

[19] D. Popescu Anastasiu and D. Radulescu, "Approximately Similarity Measurement of Web Sites," *ICONIP*, *Neural Information Processing*, Proceedings, LNCS, Springer, 9-12, 2015.

[20] D. Popescu Anastasiu and I. A. Popescu, "Model of determination of coverings with web pages for a website, *International Conference on Virtual Learning*, pp. 279-283, 2015.

[21] H.-S. Kim and S.-B. Cho, "Application of interactive genetic algorithm to fashion design," *Engineering Applications of Artificial Intelligence*, vol. 13, no. 6, pp. 635–644, 2000.