Online Generator and Corrector of Parametric Questions in Hard Copy Useful for the Elaboration of Thousands of Individualized Exams

Francisco de Assis Zampirolli[a], Fernando Teubl[b] and Valério Ramos Batista[c]
Centro de Matemática, Computação e Cognição, Universidade Federal do ABC (UFABC),

Keywords: Automatic Item Generation, Multiple Choice Questions, Parametrized Quizzes.

Abstract: In this work we present a method to produce parametric questions, which can be useful for various classes and institutions. Our method generates exams with different texts but the same content, hence the same level of difficulty. We made several interfaces in order to gather information about institutions, programmers, courses, topics, classes, exams and questions. Our method makes questions belong to topics, which in their turn belong to courses. Hereby we detail our process of producing parametric questions, which utilizes a bit of LaTeX and mainly some parts in Python. With this programming language we define the parameters inside the statement of the questions. The presented results show that we were able to build an efficient system even in its first version defined on web pages written in Django. Our system is free and it offers to teachers and professors the possibility of generating and correcting tests in an automatic and comfortable way. Since the exams are printed with different texts but the same content, then even for hundreds or thousands of students the method will be fast, effective and also efficient.

1 INTRODUCTION

Thousands of students are evaluated by teachers and professors from many branches of science every day. Evaluating students fairly is a challenging task. The problem becomes much more difficult to handle when it involves a very large number of applicants, which is the case of famous tests like TOEFL and DELE (*Diplomas de Español como Lengua Extranjera*) for language proficiency, and NAEP for educational progress. Like DELE, TOEFL examinations are taken worldwide. Numbers in (Educational 2018) indicate that it is accepted in more than 10,000 colleges, agencies and other institutions in over 150 countries. Regarding NAEP, nowadays circa 51 million students attend elementary and secondary schools in the USA, hence the number of applicants is huge (NCES National 2018).

Of course, such exams require very complex logistics to assure reliability of the results because of their continental extent. But in the case of any big educational institution we have developed an online platform that produces and corrects tests automatically even for thousands of students. They must solve questions in hard copy, and the results are reliable because though exams have all the same content their texts are different. Cheating is practically impossible since the questions are parametric, hence answerkeys do not coincide.

At the Federal University of ABC (UFABC) in Brazil we could recently validate our method through the variance in marks obtained by students from two exclusive models of the course Introduction to Programming (IP). Details are given in (Zampirolli et al., 2018) but here we present a brief description. IP belongs to an interdisciplinary undergraduate programme called Bachelor's in Science&Technology (BST). BST consists of 2,400 hours and takes three years, each one divided in three trimesters. On IP freshmen always enrol in the third trimester. After BST the student can choose many other complementary programmes like Computer Science and Aerospace Engineering.

At UFABC classes are heterogeneous because our students come from many different backgrounds. Almost 2,500 enroll on IP every year if we consider the two models: IPC (classroom) and IPH (half-distance-learning/half-classroom). In IPH only the three student evaluations must be in classrooms and the course

[a] https://orcid.org/0000-0002-7707-1793
[b] https://orcid.org/0000-0002-2668-5568
[c] https://orcid.org/0000-0002-8761-2450

Zampirolli, F., Teubl, F. and Batista, V.
Online Generator and Corrector of Parametric Questions in Hard Copy Useful for the Elaboration of Thousands of Individualized Exams.
DOI: 10.5220/0007672603520359
In *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU 2019)*, pages 352-359
ISBN: 978-989-758-367-4

is fully coordinated. This is not the case of IPC, whose greatest problem resides in a fair evaluation of its students.

IPC has in average 50 classes per year and 30 students per class. It is scheduled for five hours per week, and two of them are lab lectures. Three classes meet for each classroom lecture, where then we have 90 students. Of course, each laboratory is supervised by a single professor. For IPC they are circa 40 in total and each one has a different way of evaluating students. Hence IPC does not count on a unified examination and has a large variance in marks.

This problem does not happen in IPH, where both examination and content are fully unified. Students access all the content of IPH in a same platform called *e-learning Tidia* and developed through https://sakaiproject.org. In (Zampirolli et al., 2018) we obtained, for students that failed IP in 2017, the variance of 2.3% for IPH, which is negligible compared with 16.3% for IPC. See that reference for details on statistical analyses performed on all classes of IPC and IPH from 2009 to 2017.

These statistics must be taken as an overview because IPH differs from IPC in the following setup: in each trimester circa only 180 students enrol on the course, and in average they are supervised by just five professors but they count on teaching assistants that help the students solve 35 lists of exercises.

In IPH three exams (the first in classrooms and the others in laboratories) are all furnished by our online generator of parametric questions, a platform named *webMCTest*. We strongly believe that IPC will also present a similar variation as IPH when that course becomes coordinated. This will give more evidence that webMCTest represents a valuable means of fair evaluations.

2 RELATED WORKS

In (Zampirolli et al., 2018) we presented evaluations of students for dissertation questions carried out with a previous version of webMCTest. This and former versions are just called MCTest because they are not completely online. Parametric questions were not discussed in (Zampirolli et al., 2018) , which used MCTest 4.0, though they are already supported by this version. With webMCTest parametric questions are prepared as explained in Subsection 3.2, and more specifically we resume the method for MCTest in Subsubsection 3.2.2, which can also be found in (Zampirolli et al., 2016).

Before we draw comparisons between ours and other methods, if the reader prefers to first have an overview of webMCTest the short video at

https://youtu.be/SxQlw9ADxe8

can be quite helpful.

In (Gusev et al., 2016) the authors present a tool of online multiple choice tests for student evaluation. Their tool resorts to Information and Communication Technology (ICT). In their database, questions are grouped by content: if each three of them that treat of the same content are answered correctly, then the student is directed to the next content. Otherwise the student gets extra questions on the same subject in order to reinforce learning. Their questions are written in XML files with several tags that describe the many variations of multiple choice questions, such as weight, number of correct answers and penalty for wrong answers. Their work is similar to the one presented in (Zampirolli et al., 2018), which however differs in format (LaTeX instead of XML) and purpose (hard copy instead of online).

A more specific reference is (Del Fatto et al., 2016), which presents an ICT applied to the course Operating Systems of a master's programme. The authors programme in Bash (shell commands of Unix operating system), and in their work they present exams consisting of 45 exercises in Bash that were sent through LMS Moodle. These exams were taken in the very LMS and they made use of the available databanks of questions.

In (Kose and Deperlioglu, 2012) the authors introduce an ICT devoted to solving problems in the programming language C. This ICT can diagnose how much knowledge of C the student has, and it can also elaborate specific questions with some feedback and hints that help solve each problem. The authors developed a resource, which is an interface for the student to click-and-hold and then release in order to make up a code. This interface is however incorporated in an e-learning system. The students learn from their mistakes by means of warning messages whose model is based on restrictions. Their ICT chooses new problems for the students according to the questions they have already solved and the time taken for that. Also, there is an evaluation platform on which students can take multiple choice tests prepared according to their learning levels.

For the research area of Medical Education (Gierl et al., 2012) introduces a *cognitive model* to generate a database of multiple choice questions. Such models follow the representation of knowledge and skills that are necessary to solve a problem (Pugh et al., 2016). The study presented in (Gierl et al., 2012) begins with a specialist in Medicine creating a cognitive model to evaluate a specific topic. For each question many correct alternatives are produced. Finally, the genera-

tor programmed in Java shuffles both alternatives and question statements in order to produce the database. The result is a set of generated items with their respective answerkeys. These items are saved either in HTML or Word DOC format. Our webMCTest complements their work because the specialist can use the same cognitive model to make the database of question statements with their variation of right and wrong answers, providing the items are saved in a very basic TXT format exemplified in Subsubsection 3.2.2. In this case webMCTest will be able to correct their tests automatically.

In (Calm et al., 2013) the authors analyze the use of tests with parametric instructions in Mathematics by means of the Moodle platform with the symbolic calculator program Wiris www.wiris.com. The advantage of webMCTest is that it works with the Sympy library (www.sympy.org), which enables the inclusion of graphs of equations, and also the production of tests in hard copy, which is useful for large classes.

Except for (Zampirolli et al., 2016; Zampirolli et al., 2018) none of the works discussed in this section deal with tests in hard copy, neither to generate nor to correct them online.

3 METHOD

This section explains how we have implemented our system of evaluation of students (and applicants in general). Its main difference from other systems resides in its production of hard copy tests, which are useful when the institution cannot furnish computers for a large number of students.

WebMCTest is part of a decade-long research that began in order to meet internal demands of our institution, like a great quantity of students to evaluate. We started developing our most recent version of webMCTest in May 2018 and were able to conclude it in four months. This newest version comes with tutorial videos that help professors make use of the whole system. The most recent access report on webMCTest dates from Dec. 4, 2018: our web system has already generated 21,359 exams and performed 6,370 automatic corrections, with a total number of 138,452 questions. In the present day our databank of questions has 367 of multiple choice and 31 that are dissertative. Among them 38 are parametric, and 30 professors are already registered on our platform.

Our system was implemented in Python 3.6 with Django 2.0 and it utilizes a MySQL database server. The server runs on Ubuntu Linux 18.04.

3.1 System Structure

The main entities of our system are gathered in four components:

Programme: Institution, Programme Name, Course and Class

Topic: Topic Name, Question, Answers

Test: Exam, Student's Exam, Question for the Student

Student: Student's Data (Name, Id, E-mail)

Hence each component is a set of entities that altogether characterize the system. However, the entities "Student's Exam" and "Question for the Student" are not endowed with interfaces yet. They will be implemented in future works to include a followup of the student's records. These two entities will enable the production of personalized tests according to the student's answers in previous evaluations. The other entities all have graphical user interface (GUI) for the teacher (or professor) to input, change, remove and check data. Nowadays the main system entities are "Question" and "Exam", described as follows:

In the system each question must belong to a single topic. Each topic must be created with a bond to one or more courses. A course belongs to at least one programme. Finally, a programme is offered by an institution.

3.2 Preparing Questions from Databases

There are two means of preparing questions for a test in our system: (1) through its GUI; (2) by importing them from a TXT file in a specific format. In both cases the user must click on the link "Questions" on the left-hand side of the main page, as depicted in Figure 1.

Figure 1: Main interface of the system.

As explained at the Introduction, the main purpose of this work is to describe the production of parametric questions and the way to use them for producing a test, which can be printed in hard copy.

3.2.1 Preparing Questions Though the GUI

Click on the button "Create" illustrated in Figure 2. This opens the interface shown in Figure 3. The difficulty levels are associated to numbers from 1 (very easy) to 5 (very hard). Moreover, each question can be either dissertational or multiple choice. The user may even classify the question according to one of the five types in the *Revised Bloom's Taxonomy* (Ferraz and Belhot, 2010). Finally, the question can be parametric, so that variables in the text will be replaced with random values when our system generates the test.

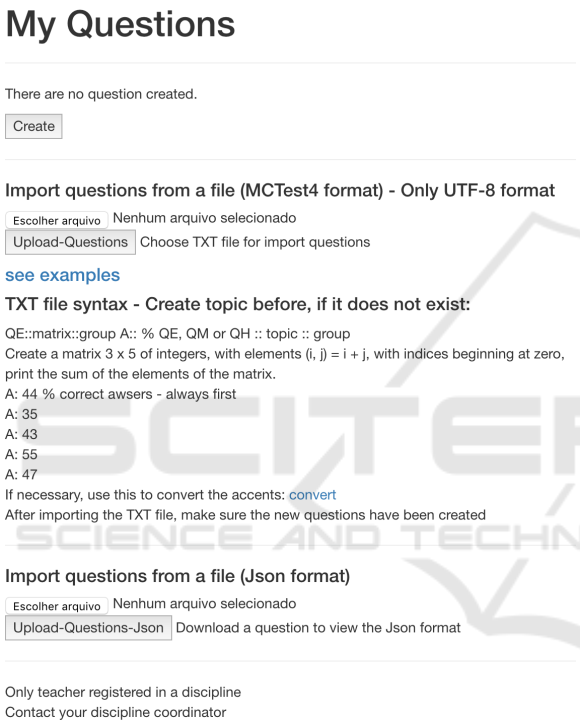


Figure 2: Interface to choose between *create* or *import* question; in the 2nd case there are instructions to format the TXT file before importing it.

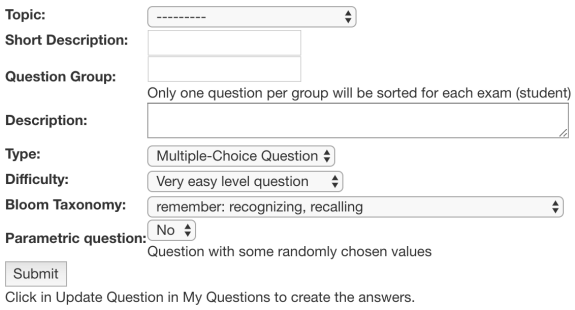


Figure 3: Interface to create a question.

In Figure 3 after choosing a topic one must give a short description and then include the statement of the question. The user will see a scrollbar with arrows named "Type", which defines the question as dissertational or multiple choice. Then it comes the level of difficulty, Bloom's taxonomy, and whether the question is parametric. After clicking on "Submit" the system shows a list of questions already created by the user. Then it will be possible to complete the question with alternatives in the multiple choice case. See Figure 4.

Figure 4: Interface to update a question.

In this second case every time you click on "Submit" there will be a new field for an extra alternative providing you have already filled out the previous one. See in Figure 5 an example of after having inserted an alternative.

On top left corner of Figure 4 there is the button "See-PDF". Clicking it makes the internet browser show the PDF for the user to check and then correct occasional errors in the format or the statement of the parametrized text. See Figure 6.

For parametric questions each click on "See-PDF" will generate a new version of the question. Figure 6 shows that the PDF specifies some characteristics of the question: Topic, Group, Short Description, Type

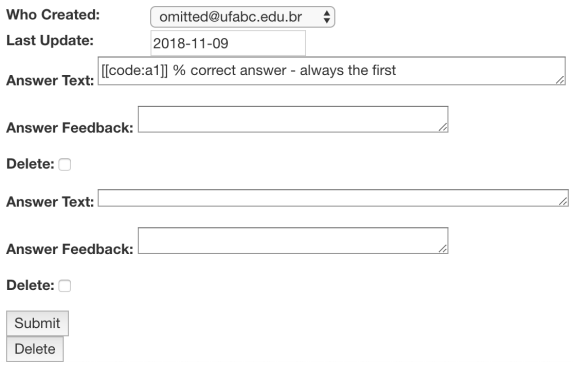


Figure 5: Part of the interface after having inserted an alternative.

webMCTest

Topic: equation-parametric
Group:
Short Description: template-equation-parametric0
Type: QM
Difficulty: 1
Bloom taxonomy: remember
Last update: 2018-11-09
Who created: omitted@ufabc.edu.br

#002 1. Write a 3×16 matrix of integers, with elements $(i, j) = i + j$. Indexes start at zero. What is the average of the last column?

A.*3 45 B.*1 33 C.*2 43 D.#0 16 E.*4 46

Figure 6: PDF generated by our system after clicking "See-PDF".

(QM for multiple choice, QT for dissertational), Difficulty (from 1 - very easy to 5 - very hard), Bloom Taxonomy, Last Update and Authorship ("Who created").

The green number indicates the register in the database. Alternatives are given by capital letters in alphabetical order from A up to J. Namely, the user can work with at most ten alternatives per question. The blue number #0 between the alternative letter D and its text means that, in the example of Figure 6, this is the right answer. The numbers in red indicate their original position in the order they were inserted (see Figure 5). This is an important feature of webMCTest because, for example, if the user inserted in the third position an absurd answer (like a fraction, an irrational or negative number, etc.), then one can identify all students that chose the absurd answer. Later in the classroom the teacher/professor can train students to make a better use of their intuition instead of just resorting to "shots in the dark".

3.2.2 Importing Questions from File

Older versions of MCTest worked with questions in TXT files. Either a user that still has these files to adapt, or even a new one that prefers not to always count on the internet server, they will both find convenient to prepare questions offline and afterwards upload them all into webMCTest with a single TXT file (for non-parametric questions). But parametric questions must each come in a separate TXT file.

This option has some other advantages: questions written in LaTeX are easily re-formatted to MCTest, and working offline spares the user from the excessive clicking of a mouse button, which is typical of online platforms.

After importing the TXT file the user will have to complement the production of questions through the graphical updating interface, described in the next subsection. This is specially important in the case of parametric questions.

The TXT file must be formatted as in the following example:

```
QE::topic::group::
statement of the question [[code:a0]] ...

[[def::
# Python code
a0 = random.random(0,5,1)
correctAnswer=111

A: [[code:correctAnswer]]
% The right answer comes first
A: 2nd alternative
...
A: last alternative
```

In this example we begin with QE. The other choices are QM, QH and QT. They indicate the respective level of difficulty (Easy, Medium, High) for multiple choice questions, whereas QT indicates a dissertation question.

Both the statement and the alternatives can include parts in LaTeX. Parametric questions have Python code inside `[[def:...]]` and `[[code:...]]`. The parameter "group" means that for each test only one question of the group will be drawn.

Back to Figure 1, after clicking on "Questions" we get Figure 2. Now one must select "Choose File" for the TXT and then click on "Upload-Questions". This automatically stores them in the system database and the user will access the whole updated list by clicking again "Questions" in Figure 1. From this point on changes and visualization are carried out as explained previously from Figures 4 to 6. In Figure 5 the user

will already find the corresponding question alternatives that came from the TXT file.

4 RESULTS

Here we present some examples of parametric questions and also how to make a test consisting of these questions.

4.1 Parametric Questions with Equations and Figure

By following the same steps shown in Figures 4 to 6 now we give an example of parametric question that includes a mathematical equation. Here it involves sinus, cosinus and integration, but one could add more complexity with other functions, free variables, limits, differentiation, etc. As explained in Subsubsection 3.2.1 the "See-PDF" button in Figure 4 will give you a new version of the parametric question whenever you click on it. Of course, in the PDF the graph of the equation will change accordingly. The question statement exemplified in Figure 4 is now more elaborate because of extra attributes, as illustrated in Figure 7. Moreover, you must choose "Yes" for "Parametric Question" in Figure 3.

```
The integrand of $[[code:a0]]$ has its graph
depicted in Figure $Integrand$. The integral function is

\begin{figure}[ht!]
  \centering \includegraphics[scale=0.5]{val_fig01}
  \centerline{Figure $Integrand$}
\end{figure}

[[def:
a1 = random.randrange(3,6,1)
a2 = random.randrange(2,4,1)
a3 = random.randrange(1,3,1)

fig = plt.gcf()
var('x')
plot(a1*sin(a2*x)-a3*cos(x),(x,0,4*pi))
fig.savefig('./tmp/val_fig01.png')

x = symbols('x')
f = a1*sin(a2*x)-a3*cos(x)
a0 = latex(Integral(f, x))
b1 = latex(integrate(f, x))
b2 = latex(integrate(f+x, x))
b3 = latex(integrate(f-x, x))
b4 = latex(integrate(x**5 + x + 1, x))
]]
```

Figure 7: Example of a parametric question with figure.

In Figure 7 notice that the parametrization resides on the coefficients `a1`, `a2` and `a3`, which take random values specified within `[[def:... ]]`. Different values will be drawn for these variables every time our system reproduces this question.

The formula `f=a1*sin(a2*x)-a3*cos(x)` of the integrand is stored in the coefficient `a0`. The integral is included in the question statement through the function `latex` of the Sympy library. Namely, the line `a0 = latex(Integral(f, x))` will be rewritten in LaTeX syntax whenever our system compiles the question. Figure 8 shows an output of the complete compilation, in which we decided to include only three alternatives according to Figure 5, so that the coefficient `b4` in Figure 7 was not included as a possible fourth alternative.

1. The integrand of $\int (4\sin(2x) - 2\cos(x))\,dx$ has its graph depicted in Figure *Integrand*. The integral function is

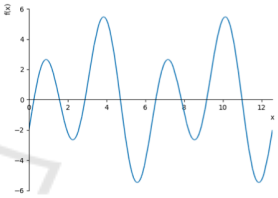


Figure *Integrand*

A. $\frac{x^2}{2} - 2\sin(x) - 2\cos(2x)$ B. $-\frac{x^2}{2} - 2\sin(x) - 2\cos(2x)$ C. $-2\sin(x) - 2\cos(2x)$

Figure 8: PDF with equations and figure generated by our system.

In Figure 8 one sees that the Sympy library uses its function `integrate` to compute the integral correctly, and webMCTest identifies it there as the alternative C. The user must assure that the other alternatives will be wrong, as indicated by `b2`, `b3` and `b4` in Figure 7.

4.2 Parametric Questions with Matrix

There are many strategies to show a matrix in the statement of a question. One of them is to include the matrix written in LaTeX, for example:

Table 1: A matrix (i, j)-entry a_0 and the elements around it.

$a_8 =$ Northwest	$a_1 =$ North	$a_2 =$ Northeast
$a_7 =$ West	$a_0 = (i, j)$	$a_3 =$ East
$a_6 =$ Southwest	$a_5 =$ South	$a_4 =$ Southeast

Another strategy is the Python library Matplotlib. Here is an example: let us define an (i, j)-entry a_0 of a matrix $M_{m\times n}$ as *West greater* in case $a_0 > \max\{a_6, a_7, a_8\}$. See part of M in Table 1, which shows only a_0 and the elements around it, namely $m \geq 3$ and $n \geq 3$. Now we write a parametric question

that makes $M_{m\times n}$ out of random integers from 0 to 9, where m and n are also random.

We can even generalize our definition as *Direction greater/lesser*, where *Direction* is one of the eight possibilities in Table 1. In our parametric question the properties "greater", "lesser" will come at random, as well as "Direction". Figure 9 shows an output of this question.

1. (35%) An entry $a_0 = (i, j)$ of a matrix is called **West lesser** if its value is lesser than the neighbouring ones positioned at a_6, a_7, a_8, as indicated in the following table. See left below an example $A = 8 \times 16$ and right below its corresponding **West lesser** entries, where "-" means "-1".

a_8 = Northwest	a_1 = North	a_2 = Northeast
a_7 = West	$a_0 = (i, j)$	a_3 = East
a_6 = Southwest	a_5 = South	a_4 = Southeast

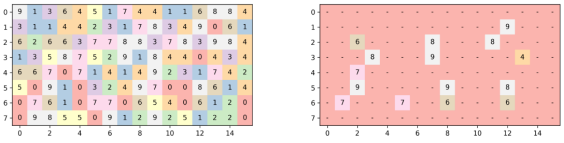


Figure 9: PDF with matrix generated by our system.

We generated the images in Figure 9 by means of the function `drawMatrix` within `[[def:... ]]`, as indicated in Figure 10.

4.3 Parametric Questions with Graph

In this last subsection we write a parametric question that includes a graph $G = (V,E)$, where $V = \{n_1, n_2, \cdots, n_p\}$ is a set of vertices and E is a set of edges such that each $e \in E$ is given by $e = (n_i, n_j)$ for $n_i, n_j \in V$. Edges can have weight and the following example solves the problem of finding the path of minimum weight. A path is a sequence of adjacent vertices, which determines a sequence of edges, and one computes the path weight by summing up the weights of its edges. See an output of this question in Figure 11.

In order to generate the image in Figure 11 we had to include the function `nx1.draw` within `[[def:... ]]`, as indicated in Figure 12.

5 CONCLUSIONS

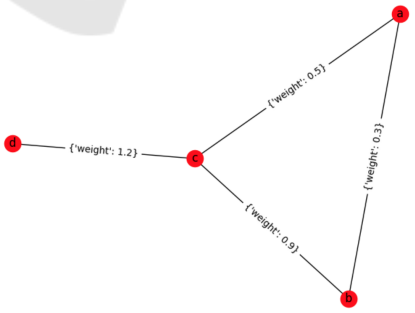
In this work we presented webMCTest, a web system devoted to generating and correcting exams automatically, in which they are made out of parametric questions. Here we gave three examples of parametric questions but their amount of types and variations is in fact directly proportional to the creativity of the user that prepares them.

Since webMCTest enables the generation of many versions of a same question, we can prepare and give them to a whole class and still be sure that

```
...
# chooses one of the 8 directions
dir=random.choice([1,2,3,4,5,6,7,8])

# nouns to appear in the table
directionAll=["North","Northeast","East","Southeast",
"South","Southwest","West","Northwest"]

# chosen direction
direction=directionAll[dir-1]

# another random value
greaterlesser=random.choice(["greater","lesser"])

# makes random matrix
m = random.randrange(7,9,1)    # number of lines
n = random.randrange(15,21,1) # number of columns
A = np.random.random((m, n))*10
A = A.astype(int)

# function to draw matrix
def drawMatrix(A,myfile):
  fig, ax = plt.subplots()
  mat=ax.imshow(A,cmap='Pastel1',interpolation='nearest')
  for x in range(A.shape[0]):
   for y in range(A.shape[1]):
    ax.annotate(str(A[x, y])[0], xy=(y, x),
    horizontalalignment='center',
    verticalalignment='center')
  plt.show()
  fig.savefig('./tmp/'+ myfile +'.png', dpi=300)
  plt.close()
...
]]
```

Figure 10: Example of a parametric question with matrix.

1. What is the Dijkstra path in this graph, between a and d nodes, with weights [('a', 'b', 0.3), ('b', 'c', 0.9), ('a', 'c', 0.5), ('c', 'd', 1.2)]?



A.#0['a', 'c', 'd'] B.*1['a', 'c'] C.*2['b', 'a', 'c', 'd']

Figure 11: PDF with graph generated by our system.

the students are all facing the same level of difficulty. Parametrized questions also require much less database storage without penalizing the randomness of the exams. Preparing them becomes a much easier task for teachers and professors.

```
What is the Dijkstra path in this graph, between
$a$ and $d$ nodes, with weights [[code:e_weighted]]

\begin{figure}[h]
  \includegraphics[scale=0.5]{fzprof_fig00.png}
\end{figure}

[[def:

import matplotlib.pyplot as plt
import networkx as nx

plt.clf()

G=nx.Graph()
e=[('a','b',0.3),('b','c',0.9),('a','c',0.5),('c','d',1.2)]
G.add_weighted_edges_from(e)

e_weighted = str(e)
out = str(nx.dijkstra_path(G,'a','d'))
out1 = str(nx.dijkstra_path(G,'a','c'))
out2 = str(nx.dijkstra_path(G,'b','d'))

plt.show()

pos=nx.spring_layout(G) # positions for all nodes
nx.draw(G,pos=pos)
nx.draw_networkx_labels(G,pos=pos)
nx.draw_networkx_edge_labels(G,pos=pos)

plt.savefig('./tmp/fzprof_fig00.png') # save as png
]]
```

Figure 12: Example of a parametric question with graph.

We have been working on many improvements to webMCTest. In the short run it will be endowed with an online corrector for the students to compare their answers with the answerkeys right after the exam. As a year long research we plan to make webMCTest work with other types of parametrized questions, which are nowadays mostly multiple choice. In the long term we are going to include methods of automatic authentication of students.

ACKNOWLEDGEMENTS

We thank the Preparatory School of UFABC for the extensive use of the webMCTest tool in order to perform three exams per year, with 600 students each. These students were selected through an exam with approximately 2,600 candidates.

REFERENCES

Calm, R., Masià, R., Olivé, C., Parés, N., Pozo, F., Ripoll, J., and Sancho-Vinuesa, T. (2013). Wiris quizzes&58; a continuous assessment system with automatic feedback for online mathematics. *Teoría de la Educación: Educación y Cultura en la Sociedad de la Información*, 14(2):452–472.

Del Fatto, V., Dodero, G., and Gennari, R. (2016). How measuring student performances allows for measuring blended extreme apprenticeship for learning bash programming. *Computers in Human Behavior*, 55:1231–1240.

Educational Testing S. (Accessed on the 28th of November 2018). Who accepts toefl scores? https://www.ets.org/toefl/ibt/about/who_accepts_scores.

Ferraz, A. P. d. C. M. and Belhot, R. V. (2010). Bloom's taxonomy and its adequacy to define instructional objective in order to obtain excellence in teaching. *Gestão & Produção*, 17(2):421–431.

Gierl, M. J., Lai, H., and Turner, S. R. (2012). Using automatic item generation to create multiple-choice test items. *Medical education*, 46(8):757–765.

Gusev, M., Ristov, S., and Armenski, G. (2016). Technologies for interactive learning and assessment content development. *International Journal of Distance Education Technologies (IJDET)*, 14(1):22–43.

Kose, U. and Deperlioglu, O. (2012). Intelligent learning environments within blended learning for ensuring effective c programming course. *International Journal of Artificial Intelligence & Applications*, 3(1):105–124.

NCES National Center E. S. (Accessed on the 28th of November 2018). Back to school statistics. https://nces.ed.gov/fastfacts/display.asp?id=372.

Pugh, D., De Champlain, A., Gierl, M., Lai, H., and Touchie, C. (2016). Using cognitive models to develop quality multiple-choice questions. *Medical teacher*, 38(8):838–843.

Zampirolli, F. A., Batista, V. R., and Quilici-Gonzalez, J. A. (2016). An automatic generator and corrector of multiple choice tests with random answer keys. In *Frontiers in Education Conference (FIE), 2016 IEEE*, pages 1–8. IEEE.

Zampirolli, F. A., Goya, D., Pimentel, E. P., and Kobayashi, G. (2018). Evaluation process for an introductory programming course using blended learning in engineering education. *Computer Applications in Engineering Education*.