TÉCNICO LISBOA

Intelligent Tutoring System for Engineering Courses

Carolina Maria Ribeiro dos Santos

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Manuel Fernando Cabido Peres Lopes

Examination Committee

Chairperson: Prof. Nuno João Neves Mamede
Supervisor: Prof. Manuel Fernando Cabido Peres Lopes
Member of the Committee: Prof. Ana Isabel Baptista Moura Santos

November 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Abstract

We present an approach for an Intelligent Tutoring System for engineering courses that generates exercises consisting of multiple choice questions (MCQs) as a way of self-assessment of the student's knowledge. Our approach provides automatic correction and feedback without the need for teacher intervention. The main contributions of this thesis are the generation of wrong answering alternatives (distractors) that address common misconceptions that students have and the generation of good quality feedback. In addition, our approach focuses on the generation of sequences of exercises with the primary goal of helping students progress faster and provide a better learning experience.

Our approach starts by generating a practice exercise of the multiple choice question type and the respective wrong-answering alternatives. Afterwards, the system generates the feedback and different instances of the same activity to allow students to have a different version of that exercise from their peers. We also present an approach to the generation of sequences of exercises that enables the developed system to select an activity for the student to practice after the completion of the current activity. The system is evaluated through two user studies in a real-world scenario with students from an Artificial Intelligence course.

We believe this system can be extremely convenient to complement the classroom and laboratory experience as it can identify gaps in the student's knowledge and help them retain information about the subject being lectured. Additionally, the system promotes self-study and alleviates the teachers' work by removing repetitive tasks and automating them.

Keywords

Multiple choice questions; Exercise generation; Automatic correction; Distractors; Feedback; Sequences of Exercises.

Resumo

Apresentamos uma abordagem para o desenvolvimento de um Sistema Tutorial Inteligente para cursos de engenharia, cujo objetivo é gerar exercícios que consistem em questões de escolha múltipla como forma de auto-avaliação do conhecimento dos alunos. Esta abordagem providencia correção automática e feedback sem ser preciso a intervenção por parte do professor. As principais contribuições desta tese são a geração de opções erradas que abordam equívocos comuns que os alunos têm, a geração de feedback informativo e com boa qualidade, e a geração de sequências de exercícios com o principal objetivo de ajudar os alunos a progredir rapidamente e providenciar uma melhor experiência de aprendizagem.

A metodologia consiste em primeiro gerar exercícios de escolha múltipla e as respetivas opções erradas. Depois, o feedback é gerado, e diferentes instâncias da mesma atividade são criadas de forma a que os estudantes possam ter uma versão diferente do exercício dos colegas. Também apresentamos uma abordagem à geração de sequências de exercícios que permite o sistema desenvolvido selecionar uma atividade para o aluno praticar depois de completar o exercício atual. O sistema foi avaliado através de dois casos de estudo num cenário real com alunos de uma cadeira de Inteligência Artificial.

Acreditamos que este sistema possa ser bastante útil para complementar as aulas e a experiência de laboratório, visto que consegue identificar falhas no conhecimento do aluno, ajudá-los a reter informação sobre a matéria a ser lecionada, promove estudo autónomo e alivia o trabalho do professor ao remover tarefas repetitivas e automatizá-las.

Palavras Chave

Perguntas de escolha múltipla; Geração de exercícios; Correção automática; Opções erradas; Feedback; Sequências de exercícios.

Contents

List of Figures

List of Tables

Acronyms

KC Knowledge Component

MCQ Multiple Choice Question

MTFQ Multiple True/False Question

1

Introduction

Contents

1.1 Context:

Online quizzes and practice exercises are becoming more common to complement the classroom and laboratory experience of students as they can help to identify gaps in knowledge and help students retain information about the subject in question. In most subjects concepts are built one upon the other and if students don't understand the first topic it becomes difficult to understand the next, and quizzes can help with this problem by using them to practice continuous self-assessment [1]. However, manually generating this type of assessment, especially if different instances for each student are desired, and correcting each question can have a high cost as it is very time-consuming. This can harm the teacher as this task requires a lot of repetitive effort better used elsewhere. If there's a high number of students taking part in these types of assessments the cost only increases which can harm the student's learning experience, and it can become impossible to write exercises for all students.

In recent years the use of online assignments and assessments has become more popular to help to solve the mentioned issues. Multiple Choice Question (MCQ)s are a prevalent form of assessment as it is a quick and effective method for assessing the student's comprehension of a given subject. This type of questions are answered by selecting the best possible answer out of a set of choices, and the questions consist of three elements [2]:

- *Stem:* a statement that introduces a problem to the student
- *Key:* the correct answer
- *Distractors:* wrong, yet plausible, answers

In [3] it was shown that multiple choice questions provide an estimate of how many students endorse correct ideas, however, there are some limitations in this type of questions. Students can have both correct and incorrect ideas at the same time regarding particular concepts which means that students who select the correct answer may still consider one or more of the distractors to be also correct. However, in this thesis, we use this type of questions as it is a quick and efficient way of assessing the student's knowledge and can help the students improve and get a better understanding of the subject that they are learning.

MCQs with constructive feedback give immediate information to the student about their (mis)understandings of the material without needing teacher intervention. This can help students overcome the difficulties of learning about a certain subject and support self-study given the fact that students can receive information about their knowledge of a certain subject on their own.

Another key aspect of Intelligent Tutoring Systems is the selection of the right activity for the student to do next. If the sequence of exercises that the system recommends to the student is efficient, it can provide a better learning experience to the students as it may help them to progress faster in their learning process.

1.2 Challenge:

If done manually, the writing of good multiple choice questions that address misconceptions with plausible answering alternatives and constructive feedback is very time-consuming and has a high cost. Also, a lot of the questions can fail to perform as intended in terms of quality when used in assessments [4].

In addition, the writing of good distractors requires training because they are the key to good quality MCQs as they can discriminate between the informed and uninformed student [5]. This discrimination is fundamental to provide feedback to the students so that they can improve, and to teachers to plan the following lessons. In addition, by identifying common mistakes made by the student we can follow their learning progress.

The generation of specific feedback for all the options of a question, both correct and incorrect, is particularly often overlooked due to the time it requires to develop [6] as it needs to have quality and be constructive to help the students understand what they need to work on to improve.

Furthermore, when it comes to the selection of the right exercise for the student to do after finishing another, a teacher can help students by proposing the activity they find most fit based on the previous problems solved by the student. However, according to [7] time resources are typically limited, which means that both students and teachers have limited time to dedicate to the exercises used as study mechanisms and their proposal. In addition, for it to be possible to develop methods for the selection of an exercise, there needs to be a definition of the complexity of each exercise. However, as mentioned, defining the complexity for all exercises used in each concept taught by the teacher can be very time-consuming, being this the process the system is trying to automate.

1.3 Objectives:

Our goal is to create a system able to generate online questions with the following properties:

- Different questions for each student but with similar levels of difficulty and quality;
- Automatic correction of the exercises;
- Plausible distractors;
- For each option, correct or incorrect, generate feedback that contributes to the student's learning;

A second objective is the automatic generation of sequences of exercises to enable the system to propose an activity to the student that makes them progress and learn faster based on the complexity defined by the sequence.

1.4 Scenario:

To accomplish the main objectives of this thesis we propose the following scenario:

- We aim at problems for which we have access to a solver, and a solver with errors.
- For each question there will be one correct choice. The other choices will consist of different errors that target common misconceptions that students might have.
- After answering a question, students receive feedback that informs which error was made if any and which exercise they should try next.
- We experiment the system in different Artificial Intelligence problems (undergraduate level).

1.5 Contributions:

There are three main contributions of this work:

- The generation of distractors that target common misconceptions of the students being tested, intending to help the student understand what they are lacking in their knowledge about the subject. The example shown in 1.5 refers to common misconceptions of students of Algebra in primary school by exemplifying errors that they can make and incorporating them into the distractors of the multiple choice question;
- The generation of feedback for each answering alternative and therefore enable the student to understand why the chosen distractor is wrong and what the correct answer is.
- We want the generated exercises to make the students' learning and progression as fast as possible, so we intend to contribute by proposing exercises according to a sequence of exercises automatically generated by the system.

Example: Simple Math Expressions

Consider the simple math expression:

$$23.4 - 2 = 21.4$$

What is the value of $23.4 - 2$?

(A) 21.4

(B) 25.4

(C) 232

(D) 236

With this example, it is possible to get familiar with several concepts involving multiple choice questions that are relevant to this work:

A – Stem As described previously, the stem is a statement that introduces a problem to the student. In this example, the stem "What is the value of $23.4 - 2$?" introduces the subtraction of two values.

B – Knowledge Component (KC) A knowledge component is the component of the subject from which the question is about that the student needs to know to be able to solve the problem correctly. In the examples' case, the student needs to know the knowledge component *subtraction*, *decimals*, *negative numbers*, and *addition* to be able to differentiate it from *subtraction*.

C – Distractors As previously stated, distractors are wrong yet plausible, answering alternatives. Our goal is for these distractors to consist of common errors that students can make and, therefore, for this specific example, we thought of the following errors:

- Mistakes subtraction with sum: $23.4 + 2 = 25.4$
- Doesn't know decimal numbers: $234 - 2 = 232$
- Doesn't know decimals and mixes subtraction with sum: $234 + 2 = 236$

Where each answer allows to identify the error made by the student.

1.6 Organization of the Document

This thesis is is organized as follows: chapter 1 introduces the subject and the challenges at hand, describes the intentions of this project and how it intends to overcome said challenges. In chapter 2 we discuss various works that addressed topics related to the ones approached in this thesis. Chapter 3 presents how we approached the described problem and the development process. Chapter 4 describes the user studies that were conducted in order to evaluate the system, the results and analysis, and discussion of these results. Finally, chapter 5 describes future work and final conclusions.

2

Related Work

Contents

In this section, we discuss several works that addressed topics of education that are relevant for this thesis. The topics consist on the generation and correction of exercises, the generation of feedback, distractors and sequences of exercises generation.

2.1 Exercise Generation

With the recent increase in popularity of the use of quizzes to assess and test the students' knowledge about the subject being taught, the generation of such exercises has become a popular research problem in the last few years. There are several ways that the generation of exercises can be done, and some of these ways are described below based on the literature that was done:

2.1.1 Generation of exercises using the Web and ontologies:

The authors of [8] present a semantic-based Automatic Item Generation (AIG) that uses Linked Open Data (LOD) and automatically generates contextual programming exercises. The tool has been used in different introductory programming courses to generate a set of practice exercises different for each student, but with the same difficulty and quality. The AIG approach is usually based on the use of test-item templates with embedded variables and formulas. The variables and formulas in the template are resolved by a computer program with actual values to generate test-items. The authors use AIG in the computer-programming domain to generate questions of different types, such as open-ended, short answer, multiple choice, and true or false. LOD is used in this semantic-based AIG approach to populate the variables of the test-item templates to generate programming exercises that are associated with real-world concepts. SPARQL was used to query linked open datasets to obtain instances of the ontological elements and use them as values for the variables of a test-item template.

The authors of [9] show an approach to generate quizzes automatically according to the official French educational standards by taking advantage of two different knowledge bases available on the Web of Linked Open Data (LOD). The two educational references of knowledge and skills are EduProgression and Les Incollables knowledge base. The selection of resources from a dataset that are relevant to the topics of a subject for a specific school year according to an educational standard is the basis for automatically generating useful quizzes for the learners, so, the extraction of a knowledge graph from DBpedia according to the educational standards was made. The extraction that was implemented enables the discovery of relevant related resources from DBpedia and it limits the number of non-relevant resources that may be added through a category-based filter. For the automatic generation of quizzes from a knowledge graph, the authors proposed an approach to automatically generate quizzes from knowledge bases on the Web of Data that relies on a different work on the generation of multiple choice questions from domain ontologies through queries but proposed a new classification of the strategies to generate quizzes. There are categories of strategies to generate quizzes and each category corresponds to a type of targeted question to be generated and to achieve it, the expected output is a special kind of RDF triple which should be retrieved from a knowledge graph by using a predefined SPARQL query template.

In [10] the authors provide a way to automatically generate MCQ tests from arbitrary domain ontologies. The authors use a Design Science Research (DSR) approach to extend the research in important ways: it introduces an open-source, web-based experimental tool for generating questions from ontologies and question stem templates, and it recommends practical guidelines for the development of similar tools based on the evaluations of generated questions. The tool was developed as a standalone web application using open-source and free development tools. The tool provides a fully-functional standalone web-based experimental system for generating MCQs from ontologies that uses templates for generating questions stems corresponding to different knowledge levels in Bloom's taxonomy (1956).

The authors of [4] present a study on the use of ontology-based MCQ generation in real education settings with the main goal of evaluating the feasibility of its use if done by instructors that don't have any prior experience in ontology development. The development of the ontology was done by first introducing the instructor to the basics of ontology development in Protégé 4, after that the instructor built an initial version of the ontology and gave feedback, and the ontology was then restructured according to the received feedback. To generate the questions the authors started by computing the pairwise similarity for all the classes in the ontology using similarity measures to vary the similarity between the key and distractor as this can make it possible to vary the difficulty of the generated questions.

In [2] the authors present a technique to generate MCQs using Wikipedia as the source of information as it can be transferred to any domain or language easily. The method consisted in first finding some available MCQs on the domain of interest from the web and forming sentences from the collected MCQs named as ”reference set”. After this, a search for potential sentences from Wikipedia was done by extracting patterns from the reference set, and sentences that contain those patterns were found. The key (correct answer) was then selected and the questions were formed from the potential sentences.

2.1.2 Generation of exercises by students:

Generating multiple choice questions is very time-consuming so, according to [6], with limited resources available it's natural to turn to unpaid student work to generate the questions, but besides this, question writing can be an effective study and learning technique. To tackle this challenge the authors explored PeerWise, an online tool to enable students to create and answer MCQs. The authors derived from the analysis of the questions written by students the following principles of good questions: ”question is from the course domain”; ”question is targeted towards a misconception”; ”question is not based on reference look up”; ”question is reasonable to solve without external systems”.

2.1.3 Generation of exercises for an e-learning framework:

The idea of [11] is to encourage students to master exercises individually so that a course can be studied totally as a self-study. Each student should have an individual assignment and those assignments have similar difficulties. The generation of a quiz consists of first defining the problem, next a template XML file is created, after that the actual XML file is created by generating some admissible data randomly using MATLAB consisting of values to populate the question, and finally, the generated values for the template file have to be specified. The order of the questions is randomized, and the XML file is then transformed into PDF, HTML, or Moodle format.

The authors of [1] developed an e-learning framework with the main goal of supporting continuous assessment, supporting different kinds of problems and not only test problems, providing teachers with feedback on student weaknesses and providing students with a friendly scenario to solve practical problems. To be able to develop a module able to support different types of exercises automatically, each student needs to have a set of exercises specifically designed for them, and therefore a method to automatically generate a great diversity of exercises is necessary. To do this, the authors suggest a base exercise, which consists of a single representation of a set of problems having three main components: the problem descriptor, the problem parameters, and the solution code; and an exercise generation module, whose purpose is to reduce the demanding task that is generating different exercises for each student by a process that automatically comes up with different versions of a base exercise.

2.1.4 My work:

Contrary to the previous work, we generate exercises manually based on the solvers we have for certain problems. This is because that we want to guarantee that the exercises address problems that are about the concepts being lectured to the students. Although the template of an exercise is written manually, each question can be parameterized automatically to change some values to different ones that still fit the context to be able to provide different exercises for each student.

2.2 Automatic Correction

To provide a better learning experience for the student and alleviate the teachers' work, automatic correction is essential. This type of correction can help with the productivity of quizzes for continuous assessment since teachers don't need to correct the exercises by themselves, which is a task that can be very time-consuming and therefore harm the continuous learning experience. From the literature that was done, there are two main procedures made to achieve automatic correction of exercises:

2.2.1 The correct answer is computed with the generation of the exercise:

In [8] and [11] it was demonstrated how the authors generated multiple choice questions. With the generation of the exercise, including the computation of all answering alternatives for each question, the key (correct answer) was also computed making it possible to have automatic correction for the questions without the need of having an instructor correct the exercise manually.

2.2.2 Exercise correction module:

The authors of [1] developed an exercise correction module by coming up with a strategy that avoids entering a specific solution for each exercise. As it was mentioned when we talked about exercise generation, in this work the authors suggested a base exercise, which consists on a single representation of a set of problems having three main components: the problem descriptor, the problem parameters, and the solution code. Since different problems are generated from a base exercise, a method that automatically corrects all of them is required to avoid individual solutions for each one. The exercise correction module adjusts the solution code that corrects the base exercise to the exercise that needs to be corrected taking into account its variable parameters.

2.2.3 My work:

Contrary to the previous work, we have access to code that solves the problem we want to generate the exercise about. Therefore, the solver provides the correct answer automatically, making it possible for the developed system to have the characteristic of automatic correction for the generated exercises.

2.3 Questions with Feedback

Feedback is a very important component when it comes to continuous assessment in learning environments since with it, the students can evaluate their progress in the learning process and if the feedback is provided automatically students can receive information on their (mis)understandings without the need for teacher intervention. There are several ways feedback can be given to the learners of a certain subject, and some of them are described below together with different types of feedback based on the literature that was done:

2.3.1 Generation of feedback with the generation of distractors:

As mentioned when we talked about exercise generation, in [10] the authors provide a way to automatically generate multiple choice question tests from arbitrary domain ontologies. The authors of this work also generated distractors, and to do this a strategy took place where feedback is already encoded in that strategy. However, the feedback only consists of informing the students if the answer is true or not true, which the authors admit is not always useful and to support the learners more substantial feedback is required, claiming that further work is needed in translating the strategies to accessible language.

2.3.2 Generation of feedback by students:

The authors of [6] explored the online tool PeerWise to enable students to create and answer MCQs. However, in addition to the generation of questions and answering alternatives by students, the authors demanded that students generate constructive answering-alternative-dependent feedback. It was claimed that simply informing the student if the answer is right or wrong is limiting, so after the analyses of the students' work three principles for the writing of good feedback were derived: "feedback is constructive"; "feedback is unique and provided for each answer alternative"; "feedback for answer alternatives does not reveal the answer".

2.3.3 Generation of feedback by academics:

In [12] it's shown an investigation on how to optimize feedback to facilitate deep learning. The VLE (virtual learning environment) presented provides a generic method for intelligent analysis and grouping of student responses that can be applied to any area of study and offers benefits such as immediate feedback, significant time-saving evaluating assignments, and consistency in the learning process. Snap-drift neural network (SDNN) provides an efficient means of discovering a relatively small and therefore manageable number of groups of similar answers, therefore an SDNN approach is proposed by the authors to analyze the students' answers and gain insights into the students' needs. The e-learning snap-drift neural network (ESDNN) is trained with students' responses to questions on a particular topic in a course. For the students to not be able to identify which questions were not answered correctly the feedback is designed by academics who are presented with groups of answers in the form of templates. The educator can easily see the common mistakes and the feedback texts are then associated with each of the pattern groupings and are composed to address misconceptions that may have caused the incorrect answers common to that pattern group.

2.3.4 Generation of feedback based on semantic web technologies:

In [13] it is suggested that the design of self-assessment should be rethought to provide personalized feedback to each learner. The authors describe their approach of a generic personalized feedback framework, integrated into a personalized web-based assessment tool using semantic web technologies, that begins from profiles of registered users to enforce personalized feedback services that provide the user positive or negative feedback reflecting their skills. If the feedback is effective, it provides the user information whether their answer is correct, and tips and stimulation that guide the learner to the correct answer. The system is helping to increase the overall learner's level when the number of positive feedback is bigger than the number of negative feedback. Having said that, information related to the learning of the user needs to be retrieved, as well as assessment resources and appropriate feedback needs to be generated. The authors used a semantic web approach, that offers tools and infrastructures for semantic representation using ontologies, to be able to find information by avoiding polysemy and reducing the number of results. Various semantic models are used: user model, metadata model, feedback model, and domain model; and the ontology is described by the Protégé open-source tool.

2.3.5 Generation of feedback with program synthesis technology:

In [14] the authors present an automated technique to provide feedback for introductory programming assignments. It provides a program synthesis technology that automatically determines fixes to the solution of the student and makes it match to a reference solution written by the instructor, and an error model language to write an error model (description of potential corrections). One of the main challenges of this research is the fact that a problem can be solved by a lot of different algorithms. To solve this, the developed tool needs the instructor to provide what the problem is by writing a reference implementation and a description of the kinds of errors students might make. The last can be specified with correction rules that define a space of candidate programs that the tool needs to search to find one that is equivalent to the reference and needs the least number of corrections possible. The authors use constraint-based synthesis technology (Sketch) to search over the large space of programs. When the synthesizer finds a solution a feedback generator extracts the choices and these are used to generate the feedback in natural language.

2.3.6 My work:

In this thesis we generate automatic feedback for each answering alternative of each question. Similar to the literature that was done, we provide feedback that informs the student if the answer is correct or not. However, since we want the feedback to be as helpful as possible, we also try to presume what is lacking in the student knowledge based on the chosen answer if wrong.

2.4 Distractors Generation

The generation of good distractors is essential as they are the key to good quality MCQs and MTFQs. The reason for this is the fact they can discriminate between the informed and uninformed student [5] and poor quality wrong alternatives can make the questions trivial. However, the writing of good distractors can be very time-consuming, therefore it's important to study various ways to generate wrong answering alternatives. Based on the literature that was done, below we present some methods used for the generation of distractors:

2.4.1 Generation of distractors using a combination of several different strategies:

In [15] it is presented an approach to automatic generation of adequate distractors to form a multiple choice question for assessing a reader's comprehension of a given article. One way to generate a distractor is to substitute a word or a phrase contained in the answer with an appropriate word or a phrase that maintains the original part of speech referred to as a target word. The approach described takes the original article and the answer as input and generates distractors as output. If the target is anything that can be converted to a numerical number or an ordinal number, which can be recognized by regular expressions based on a POS tagger, then several algorithms are devised to alter time and number, and one of these algorithms is randomly selected when generating distractors. If the target is a person, then the authors first look for different person names that appear in the article using an NE tagger to identify them and randomly choose a name as a distractor. If the target is a location or an organization, the distractors are found in the same way. For other target words, the distractor candidates are found using word embeddings with similarity in a threshold interval so that a candidate is not too close nor too different from the correct answer. However, not all distractor candidates are suitable, therefore the authors filter them out first by removing distractor candidates that contain the target word and removing distractor candidates that have the same prefix of the target word. After that, each remaining candidate is ranked.

The authors of [10] provide a way to automatically generate multiple choice questions tests from arbitrary domain ontologies. To generate the distractors for these MCQs, the developed system uses a combination of semantics and annotations-based strategies.

In [16] the focus is on automatic distractor generation given a reading comprehension article and a pair consisting of the question and its correct answer to generate long and semantic-rich distractors that should be semantically related to the question without paraphrasing the correct answer. The proposed framework consists of a hierarchical encoder-decoder to model long sequential inputs such as an article. With the implemented architecture, the model can generate grammatically consistent distractors, context and question-related.

The authors of [17] focus on the generation of distractors for English vocabulary questions. The method that was implemented was based on a previous work that generates Chinese fill-in-the-blank vocabulary questions. This work showed in its evaluation that the word2vec-based criteria to collect distractor candidates outperformed others, so it was the one implemented by the authors. Distractors candidates were collected from various sources and these candidates were then filtered following English vocabulary questions writing guidelines: ”Question options should have the same part of speech as the target word”; ”Distractors should have a word difficulty level that is similar to that of the correct answer”; ”Question options should have approximately the same length”; ”A pair of synonyms in the question options should be avoided”; ”Antonyms of the correct answer should be avoided as distractors”; ”Distractors should be related to the correct answer, or come from the same general topic”. Synonyms of the target word were filtered out because the distractors must not have the same or a very similar meaning to both the target word and the correct answer. To rank the distractor candidates the authors introduce a ranking metric that aggregates word embedding-based semantic similarity and word collocation information ranking the candidates based on the target word, reading passage, and correct answer.

2.4.2 Generation of distractors randomly:

The authors of [11] generate quizzes for an e-learning framework. An important step in the creation of these quizzes is the generation of false data to populate the parameters of the distractors for multiple choice questions in a template file. The approach consists of generating some random and admissible data by making sure that all the false answers are different from the true one.

The authors of [1] developed an exercise generation module containing various types of problems. One of the groups of problems generated was test exercises that contain multiple choice questions with more than one correct solution. For these types of questions, a random process was applied to select the set of possible answers.

2.4.3 Generation of distractors by students:

The authors of [6] explored the online tool PeerWise to enable students to create and answer MCQs. For the students to be able to write questions, they also need to be able to generate wrong answering alternatives. After the analyses of the students’ work, three principles for the writing of good distractors were derived: ”three or more answer alternatives are provided”; ”answer alternatives are plausible and linked to the misconception”; ”answer alternatives are formulated to maximize readability”.

2.4.4 Generation of distractors using ontologies:

The authors of [4] present a study on the use of ontology-based multiple choice questions generation in a real education setting. To generate plausible distractors, the developed generator uses similarity measures to choose automatically the concepts of the ontology used to create the question that share similarities with the correct answer. If the goal is to generate a difficult question the generator selects the distractors that are more similar to the key, otherwise if the goal is to generate an easy question the distractors have fewer similarities to the correct answer but still share some commonalities with it to make them plausible.

2.4.5 Generating distractors using Wikipedia:

In [2] the authors present a technique to generate multiple choice questions using Wikipedia as the source of information. A framework to generate distractors was developed that handles all the categories that the correct answer and the distractors can consist of. These categories are represented by an attribute set that contains relevant information for the generation of the distractors. This information is extracted from the tabular or structured data present on the right-hand side top of a Wikipedia page or the first sentence of the content. After this, the authors search Wikipedia for a list of related candidates from the same category, extract a set that has attribute values similar to the key, and randomly pick the required distractors from the extracted set.

2.4.6 My work:

In this thesis we generate distractors manually. We do this because we want to target common misconceptions and errors that students usually make, as we believe that this can help the student understand which parts of their knowledge about the subject they are being tested might be lacking. However, like the generation of exercises, it's possible to write a template of how we want the distractors to be and automatically compute their values based on the question that is being asked.

2.5 Sequences of Exercises

Exercise sequences determine the order in which the students perform the exercises. If this order is determined accurately, the activity proposed to the student can provide a better learning experience, as it can influence their progress speed. A teacher can help by proposing which activity they find fit for the student to practice, however, this activity can be time-consuming and it may be complex for the teacher to achieve a sequence that is adequate for all students.

The authors of [7] presented an approach that adapts to each student by providing personalised sequences of learning activities to maximise the skills acquired by students. The authors, to accomplish this, developed two algorithms that rely on the empirical estimation of the learning progress. The first is named the zone of proximal development and empirical success (ZPDES) that uses very little domain knowledge and relies on an exploration graph defined by a teacher and uses the learning progress as a reward for each activity to estimate the quality of each activity. Finally, a second algorithm named the right activity at the right time (RiARiT) was developed that is more informed about the domain to explicitly estimate the knowledge level of the students and to compute a reward for the activities.

2.5.1 My work:

Contrary to previous work, the implemented system focuses on the automatic generation of an exploration graph where each node corresponds to an activity taking into account the knowledge components required to solve each activity, without relying on the teacher's work. Each student traverses the graph starting with the least complex exercise, the root node, as determined by the implemented algorithm.

2.6 Related Work Comparison

In Table 2.1 we summarize the properties of the previously discussed works, where we can see that there are fewer studies that automatically correct exercises. We believe this is the case because correcting multiple choice takes less time than generating the actual questions, the distractors, and the feedback if we already know the answer, so most studies focused on helping instructors with the task that takes more time and therefore might lead to a more productive form of assessment.

Table 2.1: Comparison of characteristics of each work

	Exercise Generation	Automatic Correction	Feedback	Distractors Generation	Sequences of Exercises
[1]	Generates open-ended, true/false, multiple choice and multiple choice with more than one correct solution questions.	An exercise correction module was developed.	No.	Generates distractors for true/false, multiple choice and multiple choice with more than one correct solution questions.	No.
[2]	Generates multiple choice questions.	No.	No.	Generates distractors for multiple choice questions.	No.
[4]	Generates multiple choice questions.	No.	No.	Generates distractors for multiple choice questions.	No.

[6]	Students write multiple choice questions.	No.	The feedback is written by students.	Distractors are written by students for multiple choice questions.	No.
[7]	No.	No.	No.	No.	Generates personalised sequences of exercises relying on an exploration graph defined by a teacher.
[8]	Generates open-ended, short answer, multiple choice, and true/false questions.	The correct answer is computed with the generation of the exercise.	No.	No.	No.
[9]	Generates multiple choice questions.	No.	No.	No.	No.
[10]	Generates multiple choice questions.	No.	The tool provides automated feedback.	Generates distractors for multiple choice questions.	No.
[11]	Generates multiple choice and short answer questions.	The correct answer is computed with the generation of the exercise.	No.	Generates distractors for multiple choice questions.	No.
[12]	No.	No.	The feedback is designed by academics.	No.	No.
[13]	No.	No.	Generates automated personalized feedback.	No.	No.
[14]	No.	No.	Generates automated feedback.	No.	No.

[15]	No.	No.	No.	Generates distractors for multiple choice questions.	No.
[16]	No.	No.	No.	Generates distractors for multiple choice questions.	No.
[17]	No.	No.	No.	Generates distractors for multiple choice questions.	No.
My Work	Generates multiple choice questions.	The correct answer is computed by a given solver.	The system provides automated feedback.	Generates distractors for multiple choice questions.	Generates an exploration graph consisting of sequences of learning activities.

3

Approach

Contents

In this chapter, we start by explaining requirements for the writing of good online exercises. Next, we explain how each component of the system was developed in order to, when combined, create an Intelligent Tutoring System that is complete with several features that help to provide a good and productive learning environment.

3.1 Requirements

In this section we defined a set of requirements that we considered necessary for a good learning experience for the students that we tried to fulfill in the implementation of the developed system:

3.1.1 Good MCQs:

Multiple choice questions consist of three main elements: a statement that introduces a problem to the student (stem), the correct answer (key) and wrong, yet plausible, answers (distractors). To prepare good quality MCQs it's required a good knowledge of the content of the subject that is being tested and its objectives of assessment, as well as good skills in writing the items.

Taking this into consideration, to be able to generate good quality multiple choice questions we followed these principles, some of them inspired by [6]:

1. The content and difficulty of the exercise are in line with the content of the subject being tested.
2. There can't be any mistakes.
3. Must be easy to read.
4. The stem and the answering alternatives need to be clear and unambiguous.
5. The time required to solve the exercise should be similar for all exercises of the same subject.

3.1.2 Good distractors:

We generate good distractors by targeting common misconceptions and errors that students make. In other words, we generate options that consist of errors that there's a high probability of a student to make for the student to understand that the wrong option they chose is not what they believe to be true and to not make the same mistake in the future. In addition, the writing of good distractors is essential as they are the key to good quality questions due to the fact that they are able to discriminate between the informed and uninformed student [5].

In order to achieve good quality distractors, we followed these principles:

1. Avoid distractors whose sentences are too long.

2. The distractors need to be plausible and close in context to the correct option.
3. The distractors need to target common misconceptions.

3.1.3 Good feedback:

We want the feedback to be as helpful as possible for the student to be able to overcome their misconceptions. With this in mind, we inform the student which mistakes were possibly made when solving the exercise.

To be able to achieve the desired feedback, the text containing it should follow these principles:

1. The feedback is constructive and informative.
2. The feedback contains the errors the student possibly made to choose the selected option.

3.2 Components

We will now explain our approach to the generation of the exercises, the writing of the distractors and the generation of feedback. We also address how we implemented automatic correction and how we automatically generated sequences of exercises.

The developed system was implemented by following a generative approach, i.e. instead of generating an exercise, the system first solves it and then finds a way to generate distractors. We generate several solvers, corresponding to different ways in which students can make mistakes, and these become the distractors. We will now explained the process in detail.

To be able to implement a system that generates the described type of exercises, it was necessary to go through several key steps represented in Figure 3.1 and explained below:

3.2.1 Exercise Generation

This is the first step of our approach. To be able to generate instances of the exercise template, we need to have access to code that solves the problem we want to generate the exercise about. For instance, to be able to compute the value of the correct answers and values of the distractors from the questions represented in the example in section 1.5, the system needs to have access to a solver that computes these values.

To make the generation of these types of exercises as efficient as possible, the system starts by generating templates of exercises whose stem and distractors can be parameterized later to enable the generation of different instances of the same activity.

An exercise template can be generated by providing the following parameters:

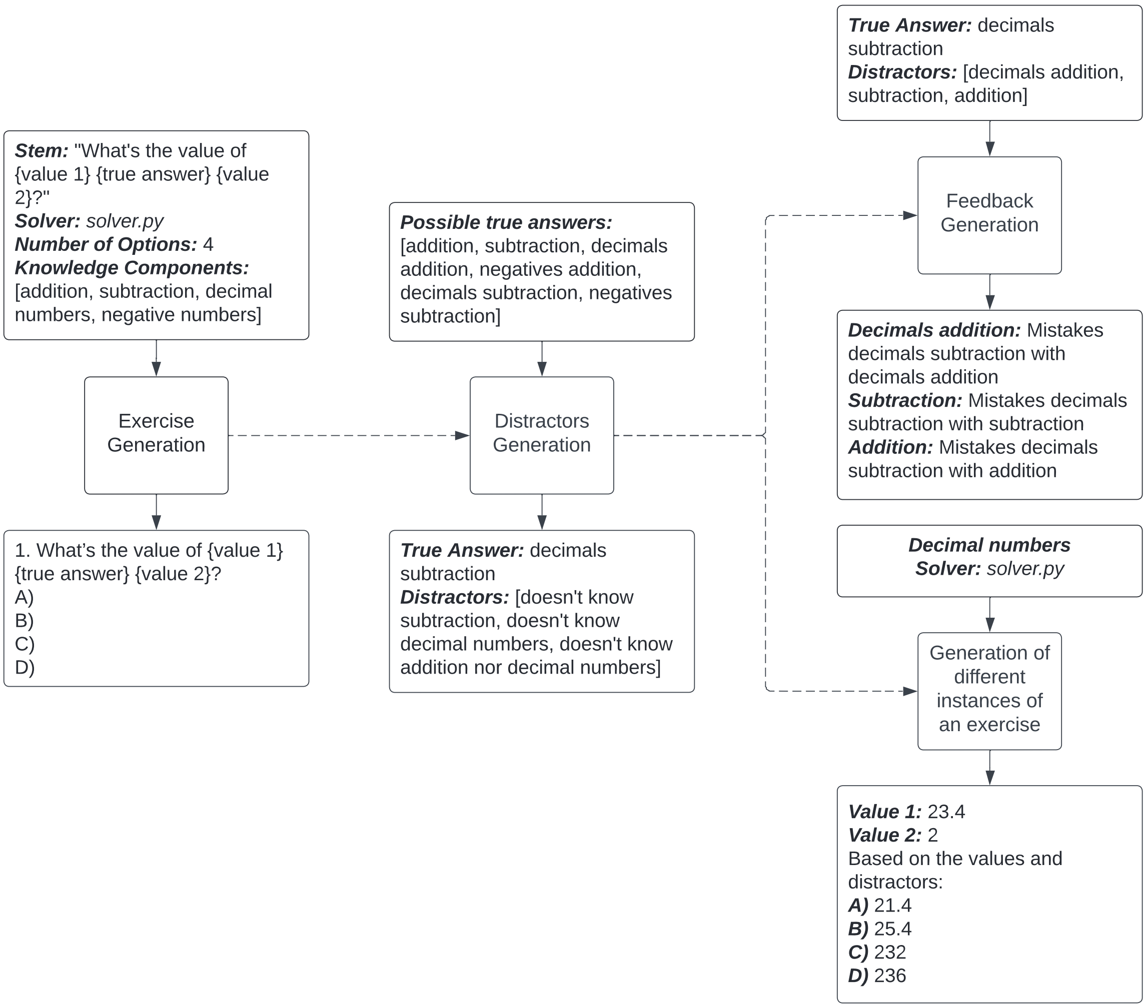


Figure 3.1: Required steps for the system to generate exercises, represented by the example in section 1.5. In order to generate the distractors, the system needs to first generate the exercise template. Afterwards, the distractors, the feedback and different instance from the same exercise can be generated.

- *Stem:* A statement that introduces the problem.
- *Solver:* A solver that provides possible solutions for the exercise and respective distractors using the knowledge components.
- *Number of Options:* The number of options of the multiple choice question.

For instance, for the example shown in section 1.5, the template consists of:

- *Stem:* "What is the value of {`value 1`} {`true answer`} {`value 2`}?", where `value 1` and `value 2` are the numbers used in the operation, and `true answer` is the subject chosen to test the student.
- *Knowledge Components:* Addition, subtraction, decimal and negative numbers.

- *Number of Options:* 4

All the mentioned parameters are manually defined, with the goal of targeting the computations contained in the provided solver, and assuring the exercise targets common misconceptions made by the students.

3.2.2 Distractors generation

We will now explain the process for the generation of distractors. However, before the system is able to generate the wrong options for an exercise, it needs one of the solutions from the solver given as a parameter to be chosen as the correct answer. This solution can be randomly chosen by the system, or given as a parameter together with the ones mentioned.

After the system computes the true answer, the distractors consist of mistakes students can make while solving the problem, i.e., errors the students can make due to the fact that they don't know the knowledge components necessary to solve the exercise and choose the answer selected to be the correct one. For instance, in the example mentioned in 1.5, the students have the following knowledge components:

(A) Has all KCs

(B) Doesn't have the *subtraction* KC

(C) Doesn't have the *decimal numbers* KC

(D) Doesn't have the *subtraction* nor the *decimal numbers* KC

All of the values corresponding to each option can be computed using the solver.

However, for some of the answers selected by the system to be correct, there were not enough mistakes to fill the number of options required. If we look at the example from 1.5, in case the number of options given as parameter was 5 instead of 4, the solver wouldn't have enough solutions to fill all the options as the number of possible combinations of lacking knowledge components isn't enough.

To solve this problem, we decided to use the solver to compute the value of all possible true answers and respective possible mistakes. For instance, if we consider that one of the possible true answers of the exercise from the example mentioned in 1.5 could be division, the system would randomly choose one of the following to fill the number of options:

- Division: 11.7
- Mistakes division with multiplication: $23.4 * 2 = 46.8$
- Doesn't know decimal numbers: $234/2 = 117$

- Doesn't know decimals and mixes division with multiplication: $234 * 2 = 468$

In other words, a possible true answer can also be considered a possible mistake as, in this case, the student can completely lack all knowledge components required to solve the problem and mistake all four basic operations of arithmetic.

It's possible that these solutions computed by the system consist of equal results. In the example, if one possible true answer is addition, the solutions will be the same as the ones from subtraction:

- Addition: 25.4
- Mistakes addition with subtraction: $23.4 - 2 = 21.4$
- Doesn't know decimal numbers: $234 + 2 = 236$
- Doesn't know decimals and mixes addition with subtraction: $234 - 2 = 232$

To solve this issue, we decided that the rest of the options that could not be filled with possible mistakes consisted of the most common values. We decided to use the most common solutions because if one value can be reached by the lack of various knowledge components, several students can choose that options despite lacking different KCs, therefore increasing the probability of that option being chosen.

3.2.3 Generation of different instances of an exercise

We wanted to automatically generate different instances of each exercise that was defined to be able to provide students with different exercises from their peers.

As mentioned, the correct answer is randomly chosen by the system, and therefore some students will have different instances of the same exercise from their peers. This is also important to allow the student to practice exercises from the same concept but with different values.

These instances can be generated by populating the stem defined during the generation of the exercise template. To be able to populate these values, the system makes it possible to define the generation of the problem. In other words, every time the system generates a new exercise, it is possible to define a new problem. For instance, a large number of exercises about Artificial Intelligence involve graphs, which means that the definition of the problem can be, for example, to randomly generate a graph, which guarantees that each student has a different exercise. When it comes to the example from section 1.5, the definition of a problem can be to randomly generate two numbers in order to populate $\{\texttt{value 1}\}$ and $\{\texttt{value 2}\}$.

The process of generating a different instance of an exercise is done inside a loop. This is due to the fact that for the values the defined problem provided, the solver might not be able to compute as many different values as the required number of options. The code from the solver can also generate exceptions, or return invalid values such as *null*. Therefore, the system will invoke the code that defines the problem until the returned values are valid options for an exercise.

In conclusion, despite the fact that the parameters from the exercise template are defined manually, the system is able to randomly generate values that can be used to parameterize the templates, and therefore provide a wide range of diverse exercises for the students to use as a study mechanism.

3.2.4 Feedback generation

When it comes to feedback generation, our main goal is to provide automatic feedback no matter which option of the questions students choose in order to immediately inform about their (mis)understandings without the need for teacher intervention. We believe that the generation of feedback promotes self-study as it can help the student understand in which areas they need to improve.

There are several types of feedback we can use for each option as follows:

1. Simply inform if the option chosen is correct or incorrect.
2. Inform if the option chosen is correct or incorrect, and if incorrect identify the correct answer.
3. Inform if the option chosen is correct or incorrect, if incorrect identify the correct answer and detect the student error.
4. Inform if the chosen option is correct, identify the correct option, detect the student error and provide the complete solution.

The feedback that is given by the system for each answering alternative is in the middle of the second and third types. This happens because we want to provide all the information in the third, but that's not possible as the solution to problems from certain subjects often involve various steps, and since students will only select the option they think it's correct the system will not see all the steps the student did to come to that answer. For instance, Artificial Intelligence problems about search strategies often involve several steps, calculations and graph traversal. However, we can still presume what is lacking in the student knowledge based on the chosen answer, being this the reason why it is somewhere in the middle of the second and third kind.

Despite the fact that the system only supports multiple choice questions, the distractors consist of common errors that students can make, and therefore we can presume what is lacking in the students' knowledge. In other words, if the student chooses an option other than the correct one, we can presume which knowledge components their lacking and need to know in order to solve the problem.

Considering that some of the distractors consist of the most common solutions resulting from various types of mistakes, the feedback consists of a list of those same possible mistakes to allow the student to identify whether the error they made is on that list and, if it is, to correct their (mis)understanding of the subject being asked.

The fourth type of feedback involves showing all the steps that are needed to reach the solution, which creates a new problem in itself: how to show the complete solution in a way that the students understand and learn from it. We will not cover this problem in this thesis.

3.2.5 Automatic Correction

We will now explain our approach to the automatic correction of multiple choice question exercises. After the generation of the previously explained components, the system provides the following data:

- Stem
- Distractors
- Feedback for each distractor
- Which of the options corresponds to the true answer (key).

We decided to present the implemented system to the students through a Web Application. This means that each exercise is presented with its respective distractors, and the student can select one of this distractors to be their answer.

Immediately after answering a question by selecting an option, the system compares the said option with the option considered to be the true answer provided by the system. The student is then able to see in the Web Application if they answered correctly, as the system will inform if their answer is the same as the true one and, if it is not, provide the feedback associated with the distractor chosen by the student, and therefore automatically correct the exercise without the need of teacher intervention.

3.2.6 Sequences of Exercises

In this section, we explain our approach to the automatic generation of sequences of exercises. Our main goal with the implementation of these sequences is to enable students to acquire different skills about a particular subject sequentially, providing a better learning experience and potentially making students progress faster in their knowledge.

As explained previously, each exercise and their distractors corresponds to different knowledge components. Our approach to the generation of sequences of exercises takes into account exercises that require the students to have the same knowledge components in order to successfully solve them. For instance, the example presented in section 1.5 asks about subtraction, however, the addition operation requires the aforementioned knowledge components. The reason for this is that the student needs to know the knowledge component subtraction to be able to understand the difference between this KC and addition.

We developed an algorithm that requires several solutions for the exercises we want the sequence to contain. These solutions consist of all options of these exercises and can have various solutions of the same exercise several times solved using different values for a more accurate sequence. For instance, if we consider the following math expression $-42.4/-2$, the file would contain the following solutions for an exercise consisting on a division of two negative numbers:

- Knows all: 21.2
- Mistakes division with multiplication: $-42.4 * -2 = 84.8$
- Doesn't know decimal numbers: $-424/-2 = 212$
- Doesn't know negative numbers: $42.4/2 = 21.2$
- Mistakes division with multiplication and doesn't know decimal numbers: $-424 * -2 = 848$
- Mistakes division with multiplication and doesn't know negative numbers: $42.4 * 2 = 84.8$
- Mistakes division with multiplication and doesn't know negative and decimal numbers: $424*2 = 848$

Taking into account that if a student doesn't know negative numbers will nevertheless answer the question correctly, if we look at an exercise consisting of a division of a negative number and a positive one, the solutions for the math expression $-42.4/2$ would be:

- Knows all: -21.2
- Mistakes division with multiplication: $-42.4 * 2 = 8 - 4.8$
- Doesn't know decimal numbers: $-424/2 = -212$
- Doesn't know negative numbers: $42.4/2 = 21.2$
- Mistakes division with multiplication and doesn't know decimal numbers: $-424 * 2 = -848$
- Mistakes division with multiplication and doesn't know negative numbers: $42.4 * 2 = 84.8$
- Mistakes division with multiplication and doesn't know negative and decimal numbers: $424*2 = 848$

In this exercise, if a student doesn't know the necessary knowledge components to correctly solve the exercise, they will not choose the correct answer. Therefore, an exercise consisting in the division of two negative numbers would be considered easier by the system than an exercise consisting on the division of a negative number and a positive one.

The algorithm returns a sequence represented by a graph. This graph is generated by computing the number of solutions of the exercises whose values are equal to the correct answer. The system considers these exercises to be less complex than the exercises whose solutions contains less common values. In other words, if the distractors have the same value as the correct answer, we assume that the student answered correctly despite lacking the knowledge components necessary to correctly complete the exercise. As a result, the exercise is deemed less complex since it doesn't require all knowledge components to be solved.

4

Evaluation

Contents

This chapter describes the methodology adapted to evaluate different components of the developed system. The data collected from the studies that were implemented in order to assess the system is presented and analysed.

4.1 Evaluation procedure and testing

To be able to evaluate if the system provides an efficient learning experience for students, we conducted two user studies:

- The first experiment consisted on assessing the quality of the feedback and if it helped students understand what was lacking in their knowledge and, therefore, progress faster.
- In the second study, we evaluate how the generated sequence of exercises helped the students to progress faster in their learning experience.

We have used the system in question to generate various Artificial Intelligence exercises to be presented to real users through a Web Application which is present in A. These users were 2nd and 3rd year undergrad students that were part of an Artificial Intelligence course offered by the Computer Science and Engineering undergraduate program at the university Instituto Superior Técnico (IST).

As stated previously in 3.2.1, to be able to generate the exercises, we need a solver to solve the problem about which we want to generate the exercise. Consequently, we decided to use the Python code for the book Artificial Intelligence: A Modern Approach as this book was used to complement the teaching in the Artificial Intelligence course. Based on the given code we manually defined various Artificial Intelligence practice exercises to access the student knowledge about the several concepts approached in the book. For example, if the generated exercise asks the student about the graph traversal algorithm $A*$, the code that contains that same algorithm provided by the solver was used to compute the answer. In addition, the solver was modified in order to contain more distractors. This was done by considering several knowledge components that the students need to have to correctly solve each algorithm and common misconceptions when solving each exercise.

Furthermore, as it was explained in 3.2.3, it's possible to define the generation of a problem in order to populate the exercise templates. For this evaluation, a possible generation was defined. This problem consisted on the generation of graphs since a lot of Artificial Intelligence problems involve them. Therefore, an algorithm that generates random graphs and trees was developed for this study, which allowed us to have thousands of different exercises to be provided to the students.

The exercises used on the studies consisted of uninformed and informed search strategies and were used as a study mechanism throughout the duration of the course.

4.2 Experiment 1 - Feedback

In this section it will be explained the process of assessing the quality of the feedback generated by the system. We start by explaining the type of exercises that were presented to the students, then we present the questions that were answered, an analysis of the data that was collected and, finally, the results of a questionnaire answered by the same students who completed the exercises.

4.2.1 Exercises

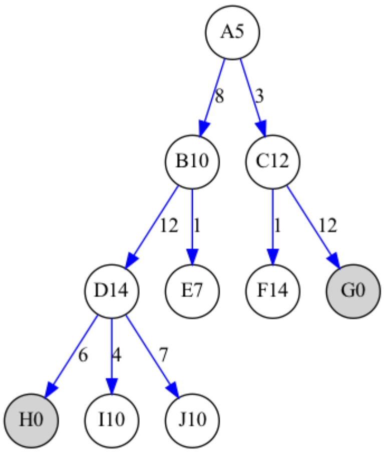
During the first experiment, the way the exercises were generated had some differences from the explanation given in section 3.2.1.

Firstly, the distractors and consequently the feedback were generated differently. The distractors, instead of consisting of different mistakes the students could make while solving the exercise, consisted of the result of solving an exercise using another algorithm as shown in Figure 4.1. The exercises were written in this format due to the fact that mistaking an algorithm such as *breadth-first search* with the algorithm *depth-first search* seemed like a plausible mistake, and even though it can happen, the students more often knew the algorithm that was supposed to be used in the exercise but got the exercise right due to mistakes made during the respective computation.

In addition to this difference, the system also generated Multiple True/False Question (MTFQ)s as represented in Figure 4.2. This questions were generated using the same methodology as multiple choice questions, however, instead of choosing one option to be the correct one, 0 or more were chosen to be true and the others were considered false. The false options would consist of the name of an algorithm, followed with the solution from another one.

During the first user study, there were in total two templates of exercises from the uninformed search subject and five templates from the informed search subject:

- Uninformed search:
  - Multiple choice question asking which nodes were expanded. One of these search strategies was chosen as the true answer and the rest were the distractors:
    - Breadth-first search
    - Depth-first search
    - Iterative deepening search
    - Uniform-cost search
  - A question similar to the previous but with the multiple true/false format.
- Informed search:



1. What is the order the nodes are expanded for the uniform-cost search? The nodes F, G are the goal states. Node A is the initial state. In case of a tie, consider the alphabetical order.

   A. **A, B, C, E, J, D, F**

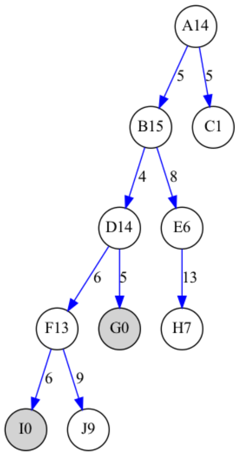
   B. A, B, C, D, G - Mistakes uniform-cost search with depth-first search.

   C. A, A, B, C, A, B, C, D, E, F - Mistakes uniform-cost search with iterative deepening search.

   D. A, B, C - Mistakes uniform-cost search with breadth-first search.

Figure 4.1: Example of a multiple choice question presented to the students during the first iteration of the evaluation of the system. Option (A) is the correct answer. The feedback shown after the rest of the options is given in case the student chooses one of them as the true answer.

- Multiple choice question asking the value of the heuristic function from multiple nodes from one of the following search strategies:
  - Uniform-cost search
  - Greedy best-first search
  - A* search
- Multiple true/false questions asking if the value of the heuristic function from one node is true or false from all the previously mentioned search strategies.
- Multiple choice question asking which nodes were expanded. One of these search strategies was chosen as the true answer and the rest were the distractors:
  - Greedy best-first search
  - A* search
  - Recursive best-first search
  - Iterative deepening A* search



1. What is the order the nodes are expanded for each search? Nodes G and I are the goal states. Node A is the initial state. In case of a tie, consider the alphabetical order. For each option indicate if it is true or false.

   A. **Greedy best-first search: A C B E H D G** - Does not know greedy best-first search.

   B. **A* search: A C B E D G** - Does not know A* search.

   C. A* search: A B C A B D E H C A B D F G - Mistakes iterative deepening A* search with A* search.

   D. Recursive best-first search: A C B E D G - Mistakes A* search with recursive best-first search.

Figure 4.2: Example of a multiple true/false question presented to the students during the first iteration of the evaluation of the system. The options in bold, (A) and (B), are the true answers, followed by the feedback given if the student considers them false. The rest are followed by the feedback given in case the student considers the options true.

– A question similar to the previous but with the multiple true/false format.

– Multiple choice question asking the value of the heuristic function from one node from the A* search strategy using the following distractors:

  * Adding all heuristics from the nodes that lead the path to the node being asked
  * Computing the value following a different path from the graph
  * Computing the value following a different path and adding all heuristics from the nodes that lead the path to the node being asked

4.2.2 Questions answered

Throughout the first study, 237 students answered the questions generated by the system. In total, students answered 17318 questions. In Figure 4.3 it is shown the number of questions answered by students. From the graph we can conclude that most students answered between 25 and 49 questions.

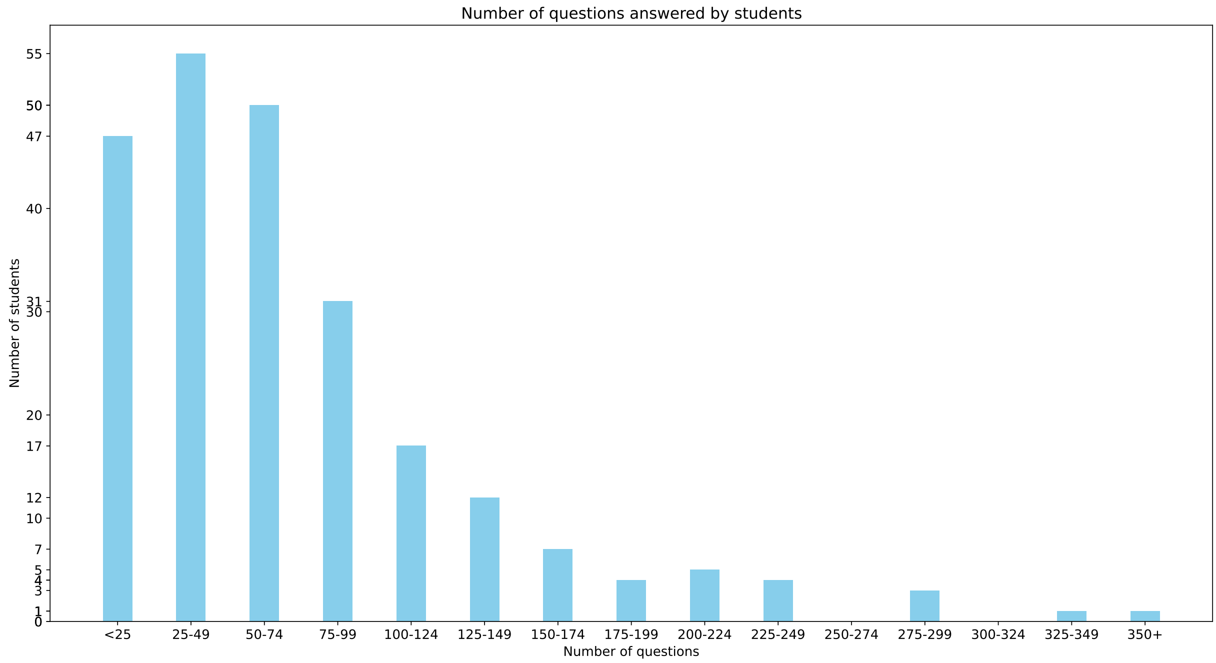


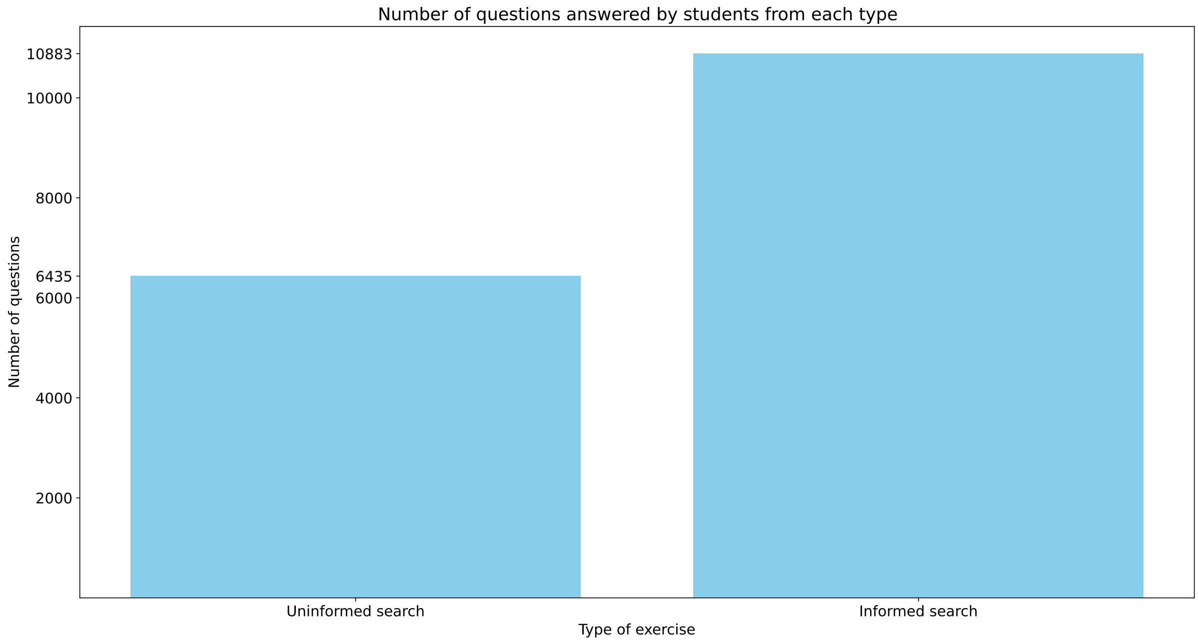
Figure 4.3: Number of students that answered the questions during the first user study.

Figure 4.4(a) represents the number of questions answered of each type. As can be observed, more questions were answered from the informed search type than from the uninformed search type. We believe this is because informed search strategies involve computations with domain-specific hints (heuristic functions) about the location of goals, and this can make the questions more complex to some students, which may lead to a slower learning progress. Additionally, the system generated more informed search exercise templates, which can motivate students to answer more questions due to the greater diversity of questions.

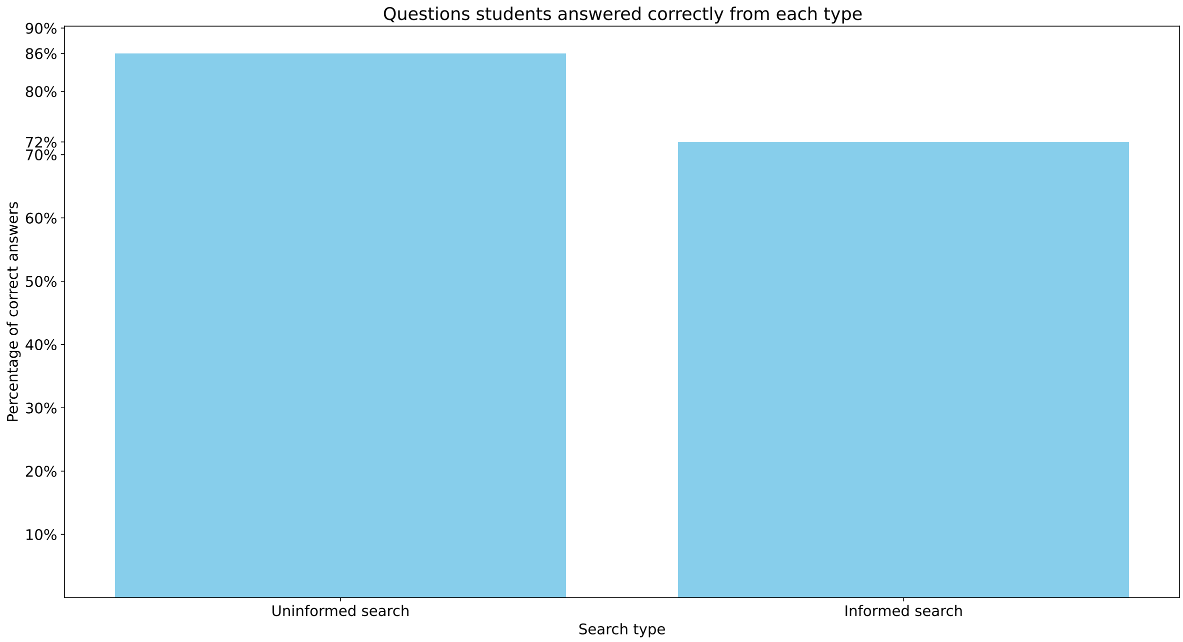
In Figure 4.4(b) the percentage of questions that students correctly answered from each type is shown. As can be observed, 86% and 72% from the questions that were answered of the uninformed search type and informed search type respectively were correct. As mentioned previously, we believe that exercises about informed search strategies may be more complex to some students, and therefore this might be one of the reasons why students answered more questions about the uninformed search type correctly.

4.2.3 Feedback quality

We assessed the quality of the feedback and if it helped students understand what was lacking in their knowledge and, therefore, progress faster. To further investigate this, we divided the students in two groups. One group would get the feedback in the first 20 questions and the other in the following 20 questions. We want to study the impact that the feedback has on the student's learning. When it comes



((a))



((b))

Figure 4.4: (a) Number of questions answered by students from the uninformed search and informed search types; (b) Percentage of questions that students correctly answered from the uninformed search and informed search types.

to questions of the informed search type, the group of students that would first receive feedback was exchanged.

Figures 4.5(a) and 4.5(b) represent the evolution of the questions that the students correctly answered from each type. The blue lines represent the group of students whose first 20 questions didn't provide feedback, and the orange lines the opposite. As we can observe, the lines represented in the graph are very close to each other, which shows that the feedback that was provided to the student did not influence the pace of their learning progress.

4.2.4 Questionnaire

To better understand the results of the evaluation, we asked students to answer a questionnaire with questions about the quality of the system, feedback, and distractors. A total of 45 students participated in this questionnaire and all considered that the system was useful for studying and understanding the subjects being taught in the Artificial Intelligence course, as shown in Figure 4.6 and Figure 4.7.

However, when students were asked to indicate from 0 to 5 how they would classify the quality of the feedback, 37.8% answered 3 as shown in Figure 4.8, and although many thought the distractors were useful to understand common mistakes, most also considered they were clearly wrong, meaning that the true answer was obvious, presented in Figure 4.9.

Furthermore, when we asked for suggestions on how we could improve the system, many students provided suggestions on how we could improve the generated questions and feedback. These suggestions included, for instance, critics to the feedback given in MTFQs since it wasn't very informative, and suggestions of exercises to include in the system.

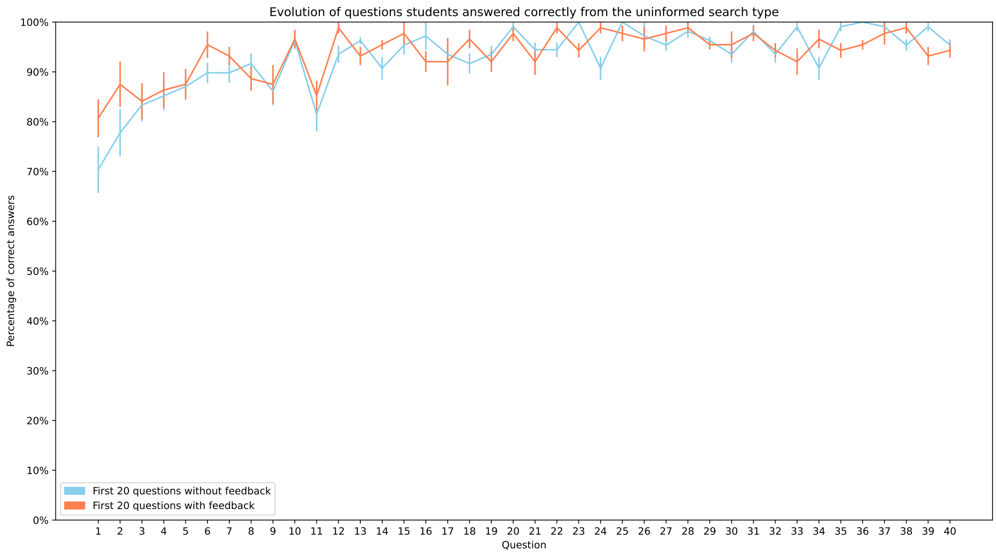
Finally, when we asked the students to rate from 0 to 5 the probability of recommending the system to their peers, most student gave a rating of 5 (53.3%) and 4 (35.6%) as shown in Figure 4.10.

4.2.5 Conclusions

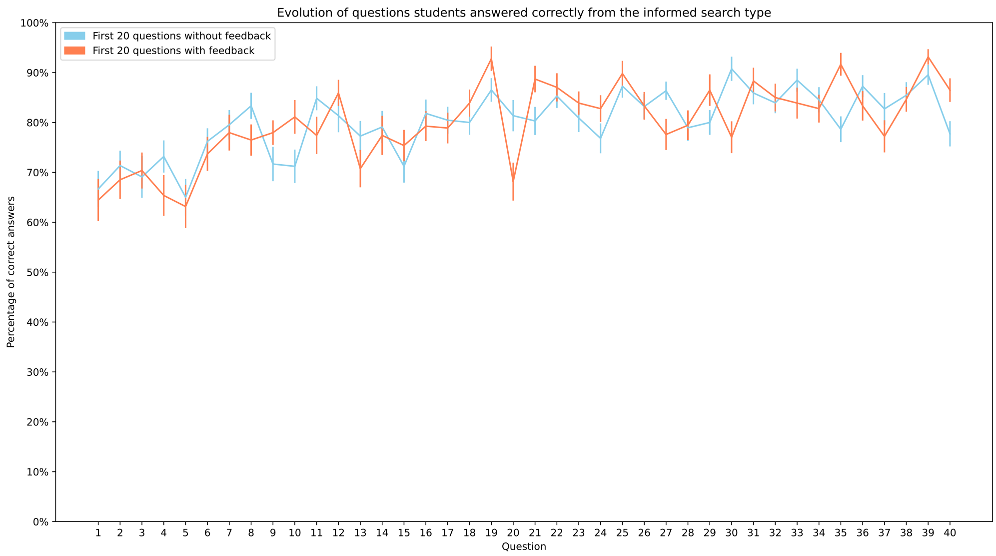
With the results from the questionnaire, we believed that the distractors and feedback highly influenced one another. In other words, if the distractors consist of specific errors that students could do when computing the solution for the questions being asked, the feedback could inform the students about that specific mistake and consequently be more specific.

After the analysis of the results, we decided to remove MTFQs from the system due to the lack of quality of the provided feedback. The students felt that the feedback generated for options such as option (A) from Figure 4.2 didn't provide enough information for the student to understand what was their mistake. This feedback is given when an option is true, but the student considers it false. However, we can't give any further information about what the student did wrong due to the fact that we didn't see how they arrived to the conclusion that the option was false.

We reached the conclusion that the best solution to this problem was to remove these MTFQs from the system, as we couldn't provide better feedback and we agreed that this type of information didn't



((a))



((b))

Figure 4.5: (a) Evolution of the questions students correctly answered from the uninformed search type; (b) Evolution of questions students correctly answered from the informed search type.

help the student to understand their mistake while solving the exercise, and therefore didn't improve their learning experience.

With this feedback, we were able to further improve the system by improving the quality of the dis-

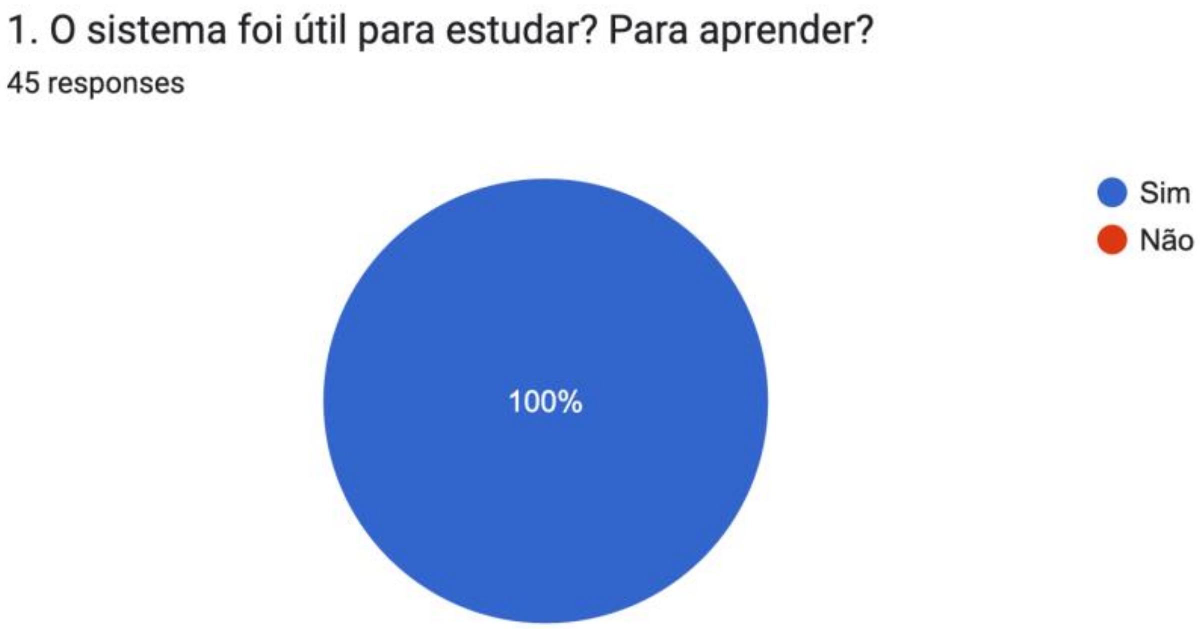


Figure 4.6: Percentage of students that considered the system useful to study.

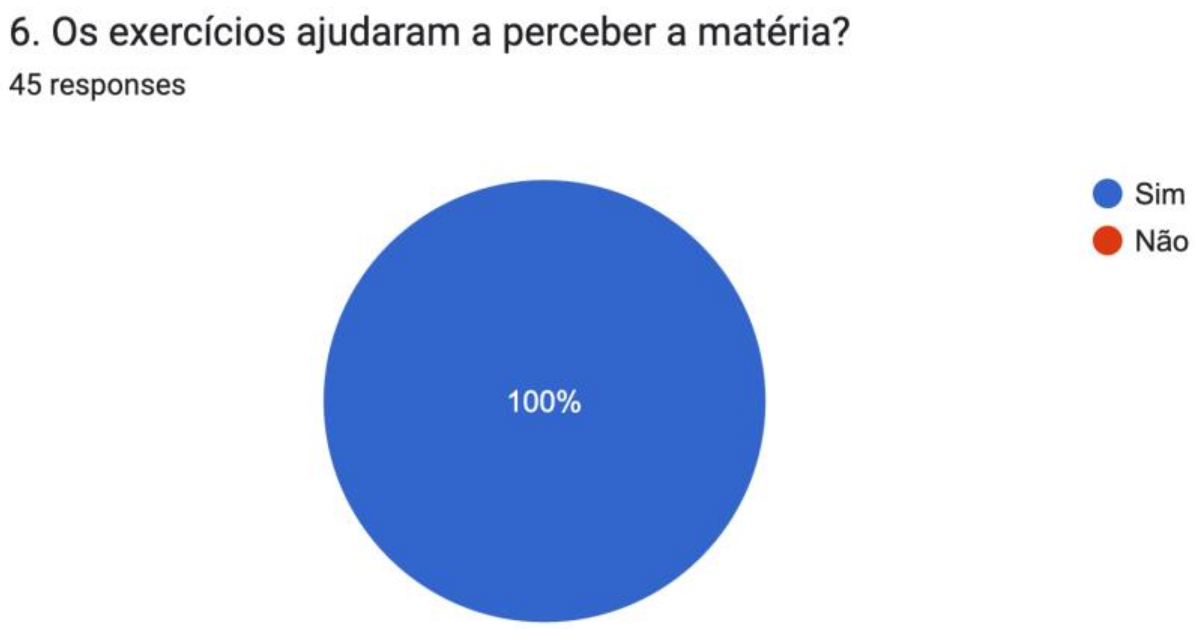


Figure 4.7: Percentage of students that considered the system helped to better understand the subjects taught in the course.

tractors and changing them for more specific mistakes that the students could make.

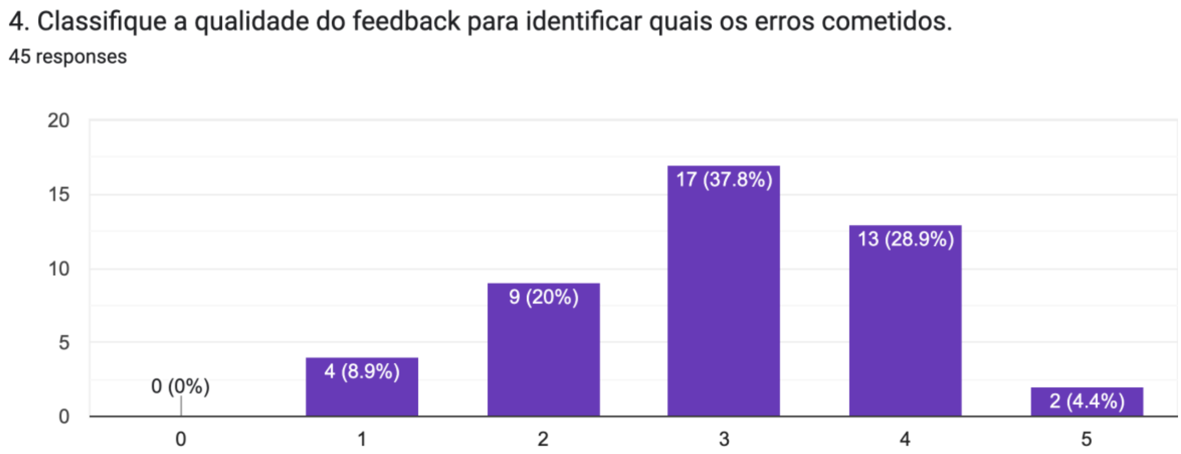


Figure 4.8: Rating from the students when asked to evaluate the quality of the feedback when trying to identify the mistake.

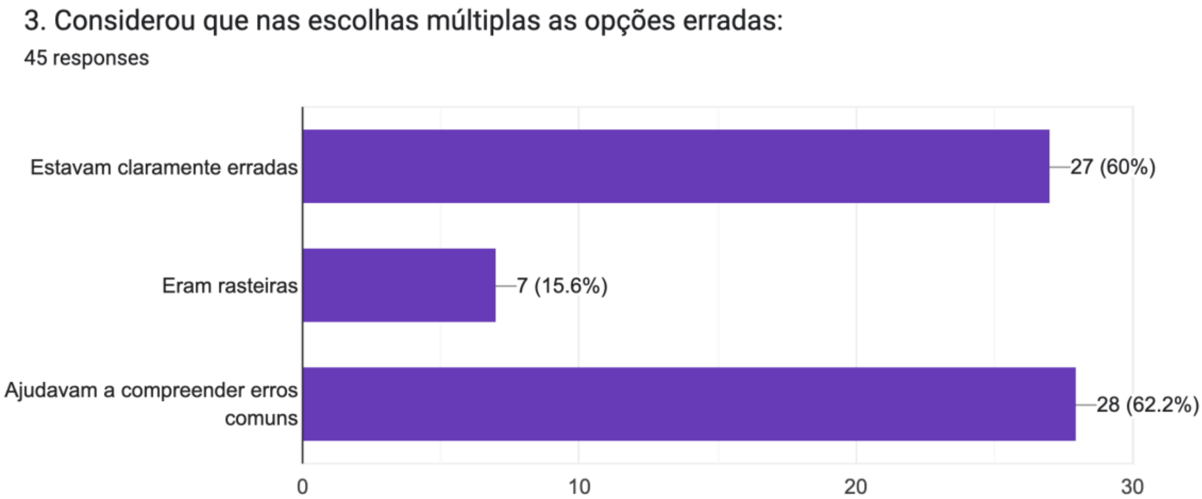


Figure 4.9: Opinion from the students when it came to the distractors. The first option is "clearly wrong", the second is "the option was tricky", and the third "it helped to understand common mistakes". Multiple options could be chosen.

4.3 Experiment 2 - Progression

In this section we demonstrate how the sequences of exercises generated by the system is evaluated on how it affects the students' learning experience. First, we present the sequences that were used to do this assessment, afterwards we present the questions that were answered, an analysis of the data that was collected and, finally, the results of a questionnaire answered by the same students who completed the exercises.

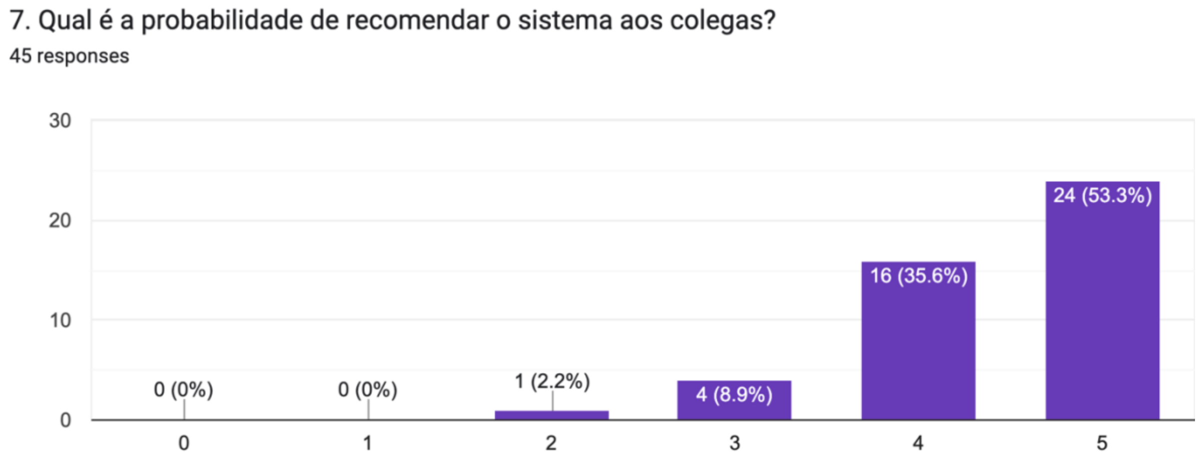


Figure 4.10: Rating from the students when asked the probability of recommending the system to their peers.

4.3.1 Exercises and sequences of exercises

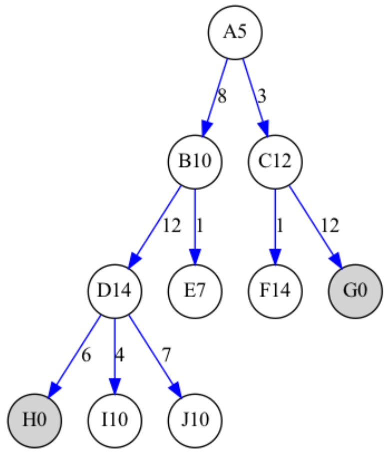
In Figure 4.11 it is shown an example of an exercise generated by the system and presented to the students during the second study. This exercise requires the student to know how to apply the *A* search*, corresponding to one of the informed search strategies referred in the book and implemented by the solver. The correct answer is option (C), and each distractor corresponds to different mistakes a student can make:

- (A) and (D): The informed search *A** uses an heuristic function consisting on the sum of the current nodes' heuristic with the distance from the current state to the initial state. Therefore, one possible mistake could be that the student might forget to use the current nodes' heuristic and only use the distance, which corresponds to solving the problem using the *uniform-cost* search as shown in option (D). In option (A) the possible mistake is identical, with the addition of instead of considering the distance between the initial state and the current state, the student only uses the distance from the previous node to the current node.

- (B): This distractor's value corresponds to two possible mistakes the student might make. The first during the computation of the heuristic function corresponding to the *A* search*, instead of only using the current state's heuristic, the student sums all heuristics from the initial node to the current node. The second mistake consists on the previous, only using the distance from the previous node to the current node.

There were in total two templates of exercises consisted of uninformed search strategies and five consisted of informed search strategies.

- Uninformed search:



1. What is the order the nodes are expanded for the A* search? The nodes H, G are the goal states. Node A is the initial state. In case of a tie, consider the alphabetical order.
    A. A, C, F, B, E, D, I, H - Possible mistakes: uniform-cost search only using the last distance.
    B. A, C, B, E, G - Possible mistakes: A* search adding all heuristics; A* search adding all heuristics and only using the last distance.
    C. **A, C, G**
    D. A, C, F, B, E, G - Possible mistakes: uniform-cost search.

Figure 4.11: Example of an exercise generated by the system during the second user study. The exercise consists of an *A* search* problem, where the correct answer is option (C), and the other options consist of distractors. If the student selects one of the distractors, the feedback presented to them is shown in front of each wrong option.

  – Multiple choice question asking which nodes were expanded. One of these search strategies was chosen as the true answer:
    * Breadth-first search
    * Depth-first search
    * Iterative deepening search
    * Uniform-cost search

    The KCs that were tested from this exercise were the same as the first iteration, except when it comes to the uniform-cost search, where it tests if the student uses all values from the path from the initial node until the current, or if they only use the value from the previous node to the current.
  – A question similar to the previous but instead asks the nodes that were generated.
• Informed search:

- Multiple choice question asking the value of the heuristic function from multiple nodes from one of the following search strategies:
  * Uniform-cost search
  * Greedy best-first search
  * A* search

  The KCs tested in this question were the following:
  * When it's an uniform-cost search or an A* search, only considers the cost from the previous node to the current
  * When it's a greedy best-first search or an A* search, adds all heuristics
  * When it's an A* search, only considers the cost from the previous node to the current and adds all heuristics
- Multiple choice question asking which nodes were expanded. One of these search strategies was chosen as the true answer:
  * Greedy best-first search
  * A* search
  * Recursive best-first search
  * Iterative deepening A* search

  When it comes to the KCs, the ones that were considered in the previous question were considered when the true answer was the greedy best-first search and the A* search. When it comes to the later two, these were the knowledge components tested:
  * Only considers the node's heuristic value
  * Adds all heuristics from the initial node to the current node without the path's cost
  * Adds all heuristics from the initial node to the current node
  * Only considers the path's cost
  * Only considers the path's cost from the previous node to the current without the node's heuristic value
  * Only considers the cost from the previous node to the current
  * Only considers the cost from the previous node to the current and adds all heuristics
- A question similar to the previous but instead asks the nodes that were generated.
- Multiple choice question asking the value of the heuristic function from one node from the A* search strategy using the following distractors:
  * Computing the value following a different path from the graph

    * Adding all heuristics from the nodes that lead the path to the node being asked
    * Computing the value following a different path and adding all heuristics from the nodes that lead the path to the node being asked
    * Only considers the cost from the previous node to the current
    * Computing the value following a different path and only considers the cost from the previous node to the current
    * Only considers the cost from the previous node to the current and adds all heuristics
    * Computing the value following a different path, adding all heuristics from the nodes that lead the path to the node being asked and only considers the cost from the previous node to the current
  – Multiple choice question where all options present 3 nodes. It asks which option has nodes where all the heuristics' values are admissible and the distractors consist of values where 0, 1 or 2 nodes have an inadmissible heuristic.

When it comes to the sequence of exercises, the system generated two different graphs. The first, represented in Figure 4.12, contains 8 exercises provided by the system for the students to solve, generated from 2 different templates: an exercise that asks the students the order by which the nodes were **expanded**, and the second that asks the order by which the nodes were **generated**. In the graph, DFS corresponds to the *depth-first search*, BFS to the *breadth-first search*, Iterative Deepening to the *iterative deepening search*, and Uniform-cost to the *uniform-cost search*.

Furthermore, the graph corresponding to the informed search strategy is represented in Figure 4.13. The sequence contains a set of 13 activities from 5 different templates: an exercises that asks the students if a node's heuristic is admissible, an exercise that asks what's the node's value computed by an evaluation function, another that asks the same as the previous but with certain particularities to the *A* search strategy*, and finally, two exercises that ask the same as the ones from the uninformed search graph but related to informed search strategies. The nodes and arrows drawn in blue represent components that were added to the graph manually. This happened due to the fact that the knowledge components tested by the nodes in blue are not tested in any other node, but as we considered these exercises valuable to test the students' knowledge these were added manually. In addition, we also manually connected the exercises corresponding to the evaluation functions to the remaining as we considered the first exercises to be essential to know to be able to successfully answer the latter.

4.3.2 Questions answered

Throughout the second experiment, 296 students used the system, and a total of 30839 questions were answered. Figure 4.14 represents the number of questions answered by students during this experiment

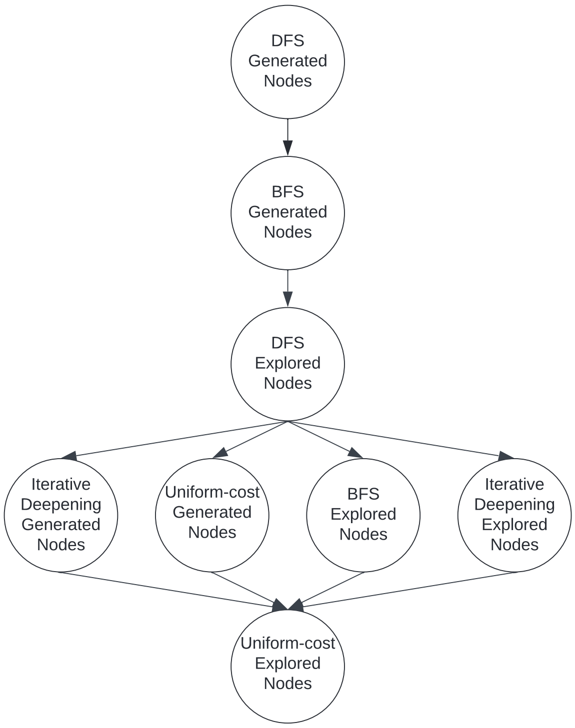


Figure 4.12: Set of activities from the uninformed search strategy subject represented by a graph from the least complex activity to the most complex one.

and, as we can observe, most students answered between 75 and 99 questions. These results show that more students used the system in the second study, and in general each student answered more questions.

Figure 4.15(a) represents how many questions from the ones that were answered are from each search strategy. We can conclude more questions were answered from the uninformed search type than from the informed search type.

In Figure 4.15(b) the percentage of questions that students correctly answered from each type is shown. As can be observed, 86% and 75% from the questions that were answered of the uninformed search type and informed search type respectively were correct, which barely differs from the previous results despite the fact that the number of questions answered increased significantly.

4.3.3 Sequences of exercises quality

In this user study, it was decided to evaluate how the generated sequence of exercises helped the students to progress faster in their learning experience. To be able to accomplish this evaluation, it was decided that the first 20 question from uninformed search answered by students with an even *id* would be chosen randomly by the system, and after these questions were answered those students would answer questions following the sequence of exercises that was generated. Furthermore, for students with an odd *id* the order of the questions would be the opposite. When it comes to questions of the informed search type, the group of students that would first follow the sequence was exchanged. The exercise presented to the student 80% of the times is the one where the student is currently situated on the sequence, and 20% one of the exercises that is immediately after in the graph. If the student correctly answer 3 times the exercises from the subject corresponding to the node they're in, the student progresses in the sequence.

Figures 4.16(a) and 4.16(b) represent the evolution of the questions that students correctly answered for each type after completing the graph. The blue lines represent the group of students whose first 20 questions didn't follow the suggestions of exercises made by the graph, and the orange lines represent the students who started with completing questions suggested by the graph. As we can observe, the lines represented in the graph are very close, which means that even though the students corresponding to the blue lines completed more questions (20 randomly chosen and the next chosen by the sequence of exercises), the students represented by the orange lines reached the same level of knowledge by just following the graph generated by the system.

4.3.4 Questionnaire

We asked the students to answer a questionnaire with questions similar to the ones asked previously. A total of 58 students participated in this questionnaire and all considered that the system was useful for studying as shown in Figure 4.17. In Figure 4.18 we can see that most students found that the system helped them understand the subjects being taught in the Artificial Intelligence course.

However, when students were asked to indicate from 0 to 5 how they would classify the quality of the feedback, the results were still lacking, as 37.9% answered 3 as shown in Figure 4.19. In this study, the results from the distractors improved as most students thought the distractors were useful to understand common mistakes as presented in Figure 4.20.

When asked to give suggestions on how to improve the system, most students mentioned the correction of certain bugs, as the website had some problems due to the amount of users using it at the same time close to the course's evaluation period, and the addition of exercises from other topics taught in the course.

Finally, when we asked the students to rate from 0 to 5 the probability of recommending the system to their peers, most student gave a rating of 5 (51.7%) and 4 (29.3%) as shown in Figure 4.21, being these results very similar to the ones from the first evaluation.

4.3.5 Conclusions

After reflecting about what's necessary to provide good quality feedback, we came to the conclusion that even thought we can presume what the student is lacking through the distractor that was chosen, we can't fully understand what was the mistake in the student's thought process when solving the exercise. These exercises consisted of Artificial Intelligence questions that required the knowledge of algorithms that traverse trees and graphs and, furthermore, involve computations and sometimes recursive thought that the system doesn't see the students doing. In other words, we believe that the quality of the feedback not only depends in the quality of the distractors, but also on the subject and concepts being asked.

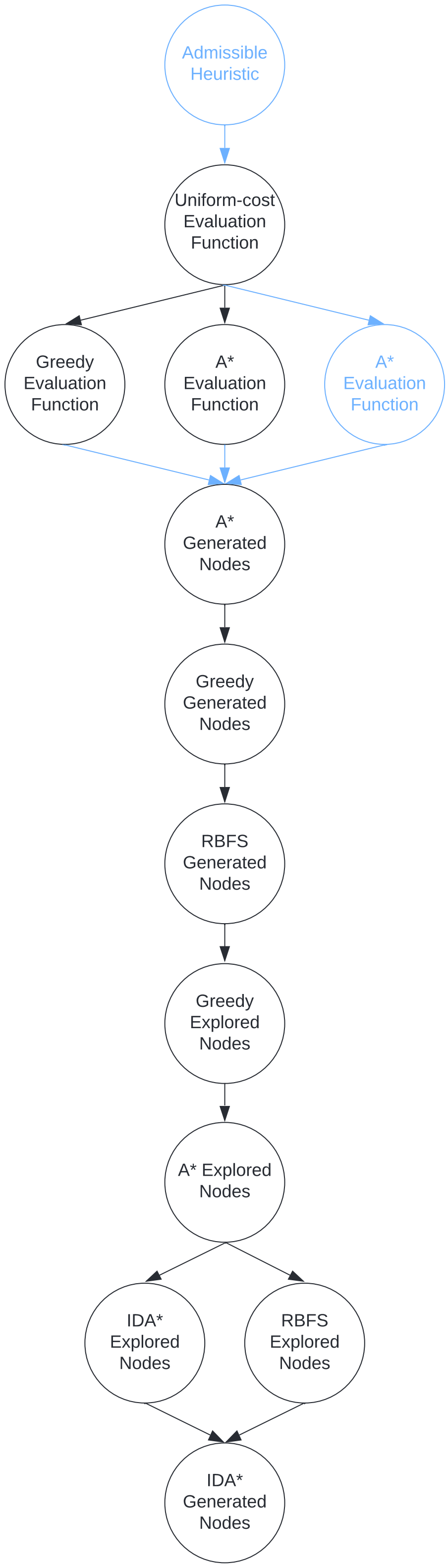


Figure 4.13: Set of activities from the informed search strategy subject represented by a graph from the least complex activity to the most complex one. The blue nodes and links are activities and connections that were added manually.

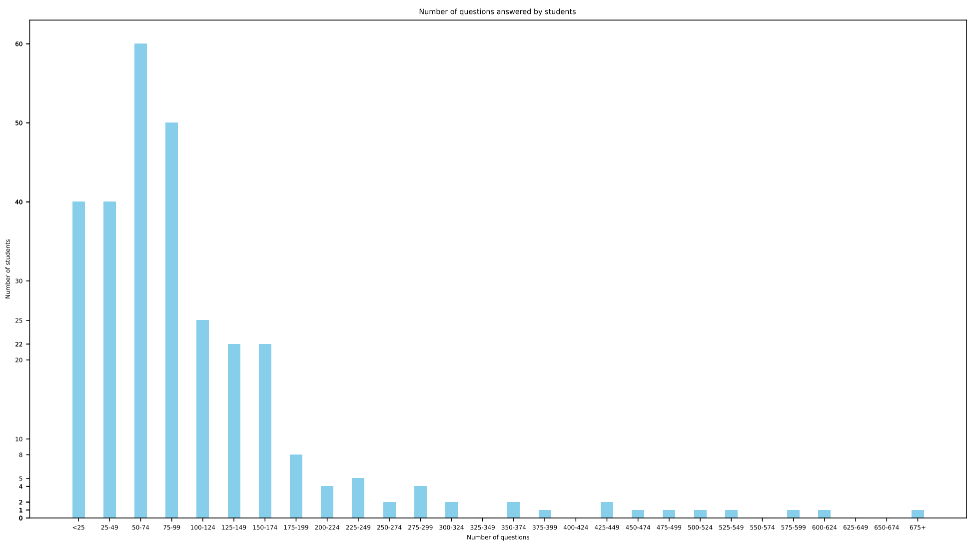
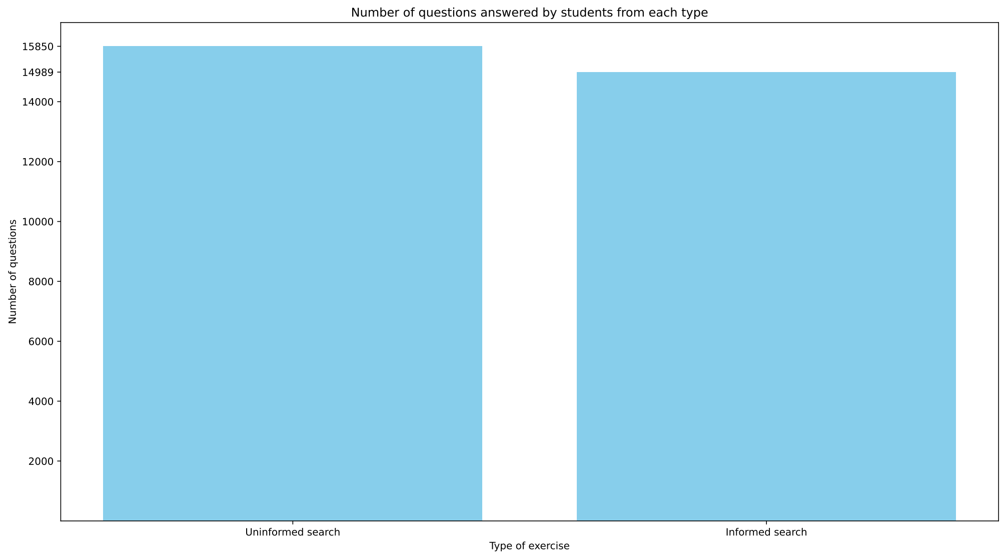
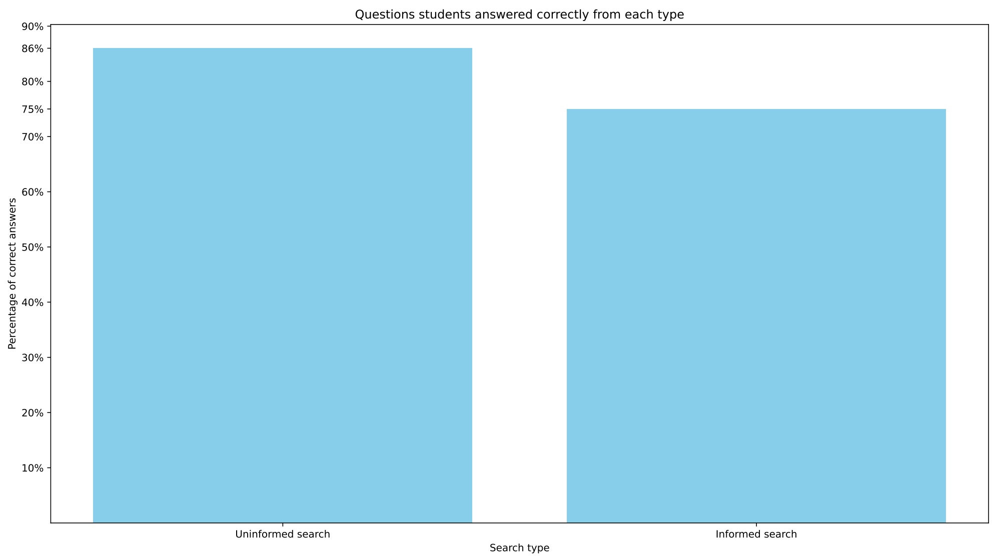


Figure 4.14: Number of students that answered the questions during the second user study.

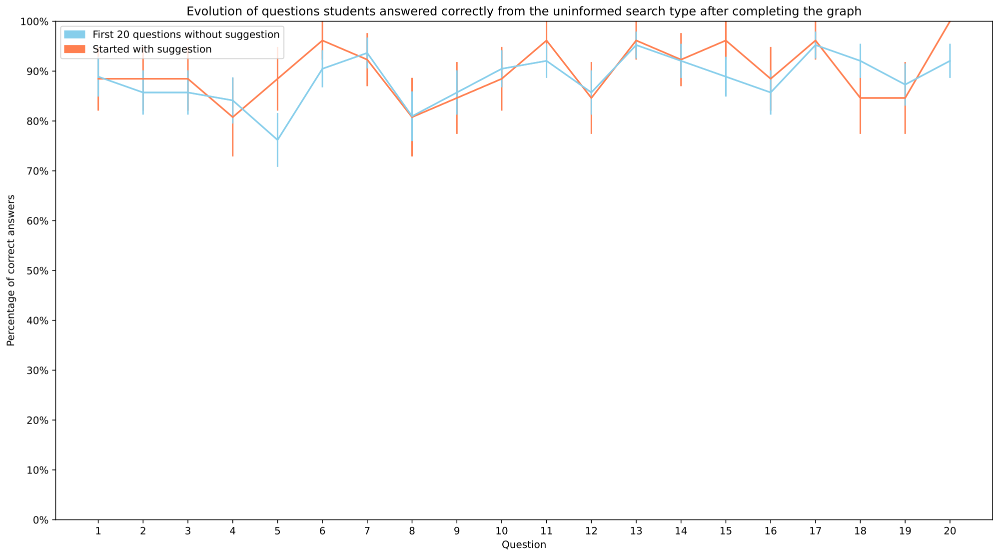


((a))

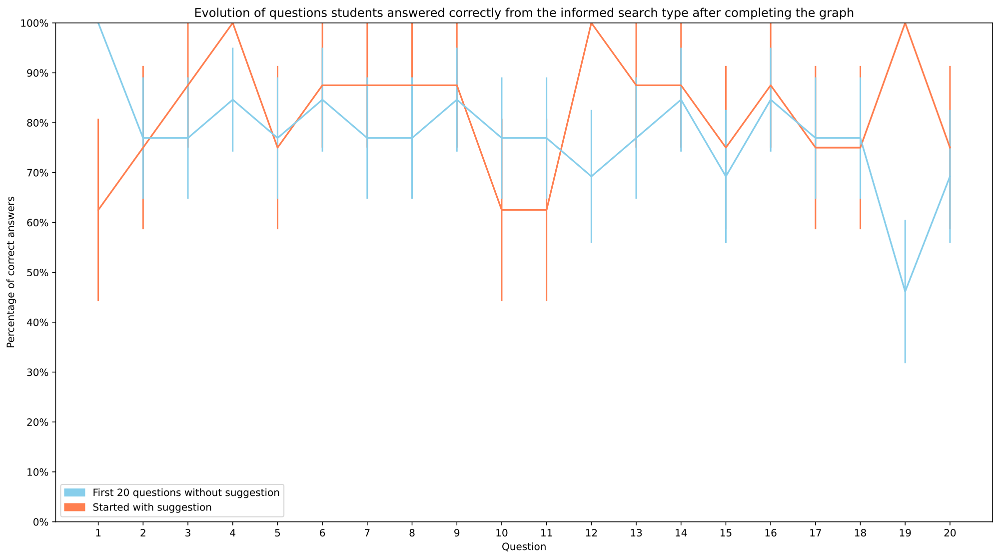


((b))

Figure 4.15: (a) Number of questions answered by students from the uninformed search and informed search types; (b) Percentage of questions that students correctly answered from the uninformed search and informed search types.



((a))



((b))

Figure 4.16: (a) Evolution of questions students answered correctly from the uninformed search type after completing the graph; (b) Evolution of questions students answered correctly from the informed search type after completing the graph;.

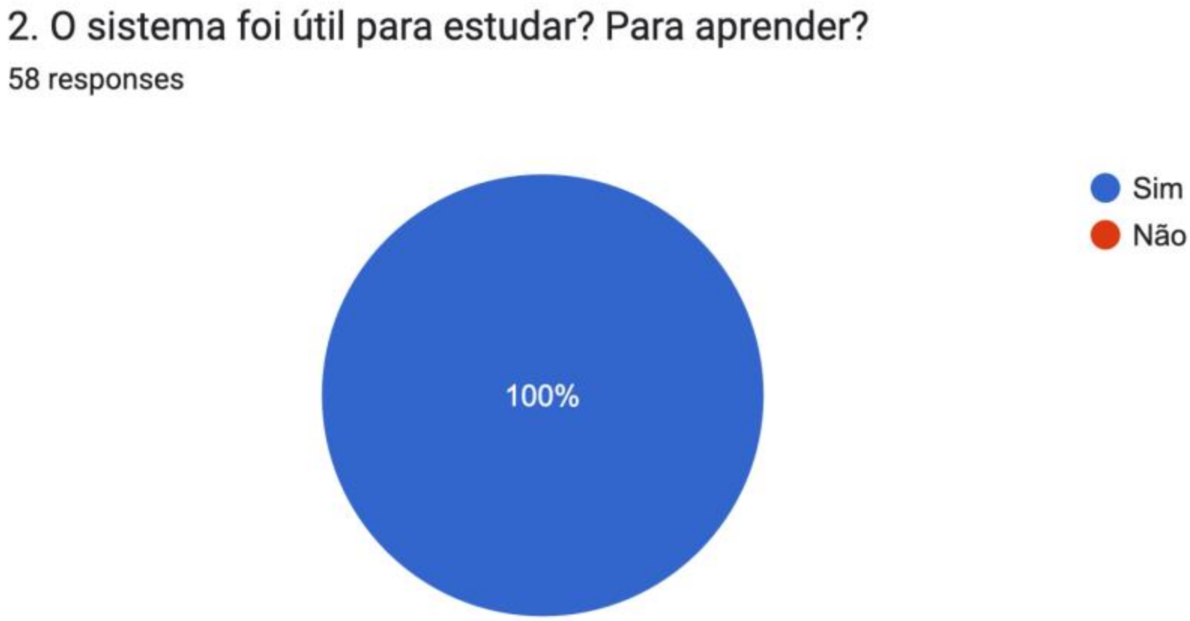


Figure 4.17: Percentage of students that considered the system useful to study.

Figure 4.18: Percentage of students that considered the system helped to better understand the subjects taught in the course.

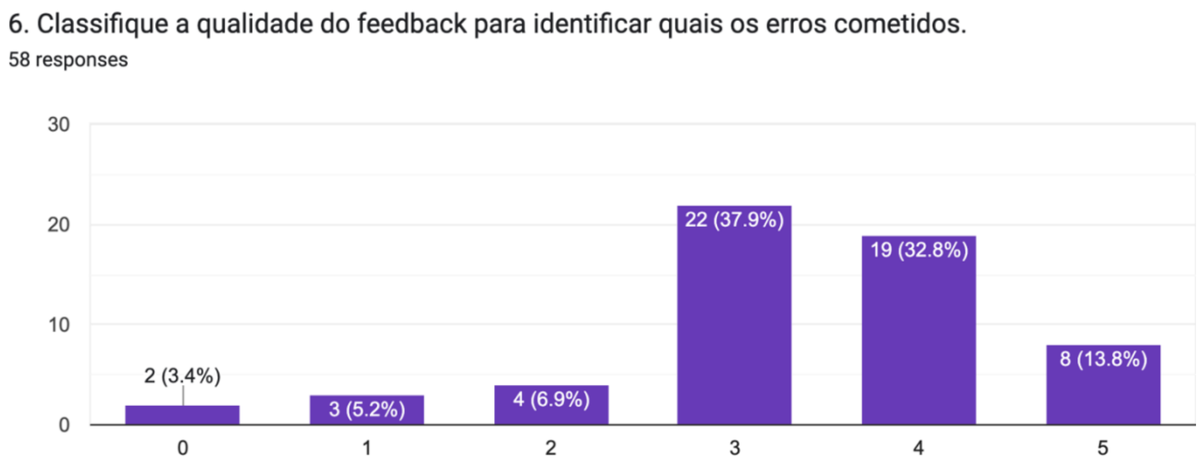


Figure 4.19: Rating from the students when asked to evaluate the quality of the feedback when trying to identify the mistake.

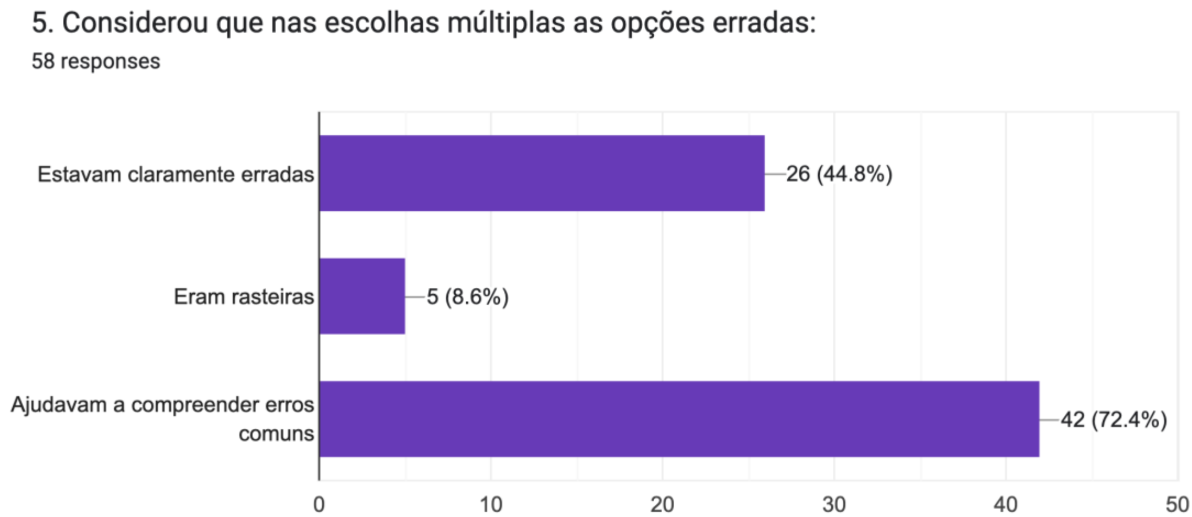


Figure 4.20: Opinion from the students when it came to the distractors. The first option is "clearly wrong", the second is "the option was tricky", and the third "it helped to understand common mistakes". Multiple options could be chosen.

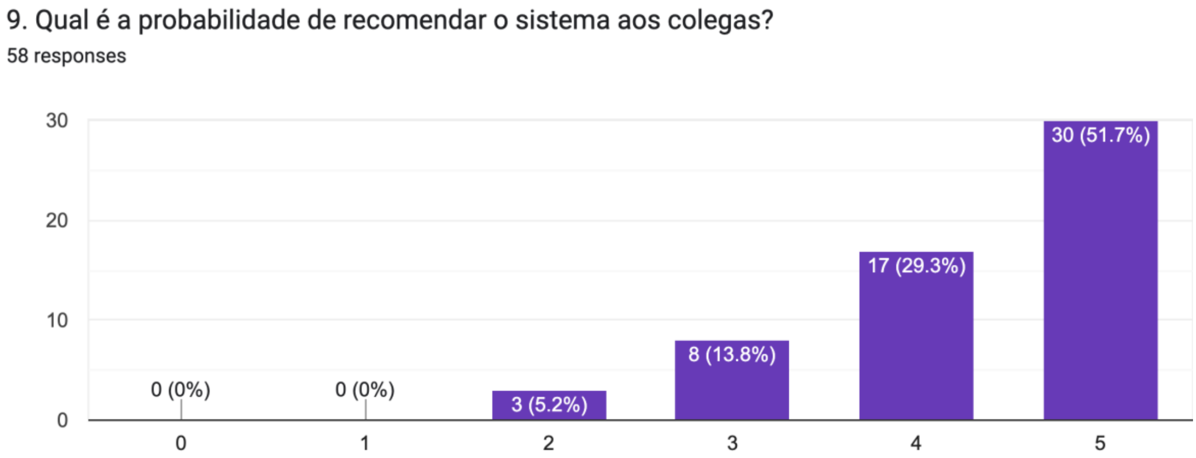


Figure 4.21: Rating from the students when asked the probability of recommending the system to their peers.

5

Conclusion

Contents

5.1 Conclusions

This work aimed to implement a system able to generate multiple choice questions (MCQs) automatically as these types of exercises are a quick and effective way of assessing the student's knowledge. In addition, our goal was to automatically correct this exercises and provide feedback.

The wrong answering alternatives (distractors) of the generated exercises consist of common misconceptions that students have, and therefore create more realistic alternatives as they consist of errors that students usually make. In addition, these distractors allow to identify the student's difficulties and, by identifying these misunderstandings, it's possible to provide informative and constructive feedback.

The mentioned feedback is given for every answering alternative, and informs the student if the chosen option is correct or incorrect and, if incorrect, identifies the correct answer and tries to presume the mistakes the student made while solving the exercise through the chosen distractor.

The system creates an exercise template with the mentioned properties, and generates various instances from that template in order to provide thousands of different exercises to the students. These exercises can be presented to the students through a Web Application, giving the system the possibility of having automatic correction.

In addition, the system focuses on the generation of an exploration graph that represents a sequence of exercises the students can follow in order to progress faster in their learning experience. In other words, each student traverses the graph starting with the least complex activity as determined by the system, and as they complete each practice exercise the system will recommend more complex exercises in order to provide the right activity at the right time.

The system was evaluated by real-world users through two user studies. The first consisted on assessing the quality of the feedback, and the second how the generated sequence of exercises helped the students to progress faster. In the first experiment, a total of 237 students participated in the study by solving 17318 questions. In the second, 30839 questions were answered by a total of 296 students.

The results of the system assessment have proven to be mixed. When it comes to the feedback, the quality was evaluated as average, however we believe this may be because the exercises used for the evaluation were Artificial Intelligence exercises, whose solving requires a considerable amount of steps that are not presented to the system due to the type of exercises being multiple choice questions. However, during the questionnaire the system proved to be promising, as all students said it helped them study and understand the subjects being lectured. Furthermore, the evaluation of the students progression using the generated learning graph also showed very promising results. Considering these results, improving the feedback is required to improve the system in the future.

We believe this system can be very helpful to complement the classroom and laboratory experience as it can identify gaps in the students' knowledge, help them retain information about the subject being lectured, and promote self-study as it provides exercises that are automatically corrected without the need for teacher intervention.

5.2 Future work

This system is the combination of several ideas and features previously explored, where new technologies are always emerging giving room for improvements and additional research. Some of the possible improvements that could be made are listed below:

- Automatic generation of the exercise stem and distractors. Although the system is currently capable of generating several instances of one exercise template, and therefore generates different stems and distractors, it still needs a template. This template could be generated automatically through, for example, the use of knowledge bases.
- In order to further improve the generated feedback, the system could provide the complete solution. This feature can be challenging, due to the fact that in several subjects such as Artificial Intelligence it can be complex to find a away to show the complete solution and the several steps that the resolution involved.
- The system could include user profiles that could be accessed both by the student (the user) and teachers. This profile could include data such as which exercises subjects the student got more questions wrong, the knowledge components the student is lacking, and many more statistics that could help both the user and the teacher to identify the students' main difficulties and therefore help in their learning progress.
- When it comes to the sequences of exercises, the system currently presents an exercise to the student if the student correctly answered the previous exercise in the sequence 3 times or 20% of the times. It could be interesting to use or implement an algorithm such as the one developed in [7] in order to better adapt the sequence to the students knowledge and capabilities.

Bibliography

[1] F. Prados, I. Boada, J. Soler, and J. Poch, "Automatic generation and correction of technical exercises," *International Conference on Engineering and Computer Education: ICECE*, vol. 5, 2005.

[2] A. S. Bhatia, M. Kirti, and S. K. Saha, "Automatic generation of multiple choice questions using wikipedia," in *Maji P., Ghosh A., Murty M.N., Ghosh K., Pal S.K. (eds) Pattern Recognition and Machine Intelligence. PReMI 2013*, ser. Lecture Notes in Computer Science, vol. 8251. Springer, Berlin, Heidelberg, December 2013, pp. 733–738.

[3] C. E. Brassil and B. A. Couch, "Multiple-true-false questions reveal more thoroughly the complexity of student thinking than multiple-choice questions: a bayesian item response model comparison," *International Journal of STEM Education*, vol. 6, no. 16, May 2019.

[4] T. Alsubait, B. Parsia, and U. Sattler, "Generating multiple choice questions from ontologies: How far can we go?" in *Lambrix P. et al. (eds) Knowledge Engineering and Knowledge Management. EKAW 2014*, ser. Lecture Notes in Computer Science, vol. 8982. Springer, Cham, April 2015, pp. 66–79.

[5] I. Burud, K. Nagandla, and P. Agarwal, "Impact of distractors in item analysis of multiple choice questions," *International Journal of Research in Medical Sciences*, vol. 7, no. 4, pp. 1136–1139, April 2019.

[6] R. Glassey and O. Bälter, "Put the students to work: Generating questions with constructive feedback," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, October 2020, pp. 1–8.

[7] B. Clement, D. Roy, P.-Y. Oudeyer, and M. Lopes, "Multi-armed bandits for intelligent tutoring systems," *Journal of Educational Data Mining*, vol. 7, no. 2, pp. 20–48, June 2015.

[8] L. Zavala and B. Mendoza, "On the use of semantic-based aig to automatically generate programming exercises," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. Association for Computing Machinery, February 2018, pp. 14–19.

[9] O. Rodríguez Rocha and C. Faron Zucker, "Automatic generation of quizzes from dbpedia according to educational standards," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18. International World Wide Web Conferences Steering Committee, April 2018, pp. 1035–1041.

[10] M. Cubric and M. Tosic, "Design and evaluation of an ontology-based tool for generating multiple-choice questions," *Interactive Technology and Smart Education*, vol. 17, no. 2, pp. 109–131, February 2020.

[11] M. Fikar, M. Bakošová, T. Hirmajer *et al.*, "Automatic generation of assignments and quizzes in control engineering education," in *2007 European Control Conference (ECC)*. IEEE, July 2007, pp. 2714–2720.

[12] R. Guo, D. Palmer-Brown, S. W. Lee, and F. F. Cai, "Intelligent diagnostic feedback for online multiple-choice questions," *Artificial Intelligence Review*, vol. 42, pp. 369–383, August 2013.

[13] L. Cheniti Belcadhi, "Personalized feedback for self assessment in lifelong learning environments based on semantic web," *Computers in Human Behaviour*, vol. 55, pp. 562–570, February 2016.

[14] R. Singh, S. Gulwani, and A. Solar-Lezama, "Automated feedback generation for introductory programming assignments," *SIGPLAN Not.*, vol. 48, no. 6, pp. 15–26, June 2013.

[15] C. Zhang, Y. Sun, H. Chen, and J. Wang, "Generating adequate distractors for multiple-choice questions," in *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KDIR*, vol. 1, INSTICC. SciTePress, October 2020, pp. 310–315.

[16] Y. Gao, L. Bing, P. Li, I. King, and M. R. Lyu, "Generating distractors for reading comprehension questions from real examinations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, July 2019, pp. 6423–6430.

[17] Y. Susanti, T. Tokunaga, H. Nishikawa, and H. Obari, "Automatic distractor generation for multiple-choice english vocabulary questions," *Research and Practice in Technology Enhanced Learning*, vol. 13, no. 15, October 2018.

A

Web Application

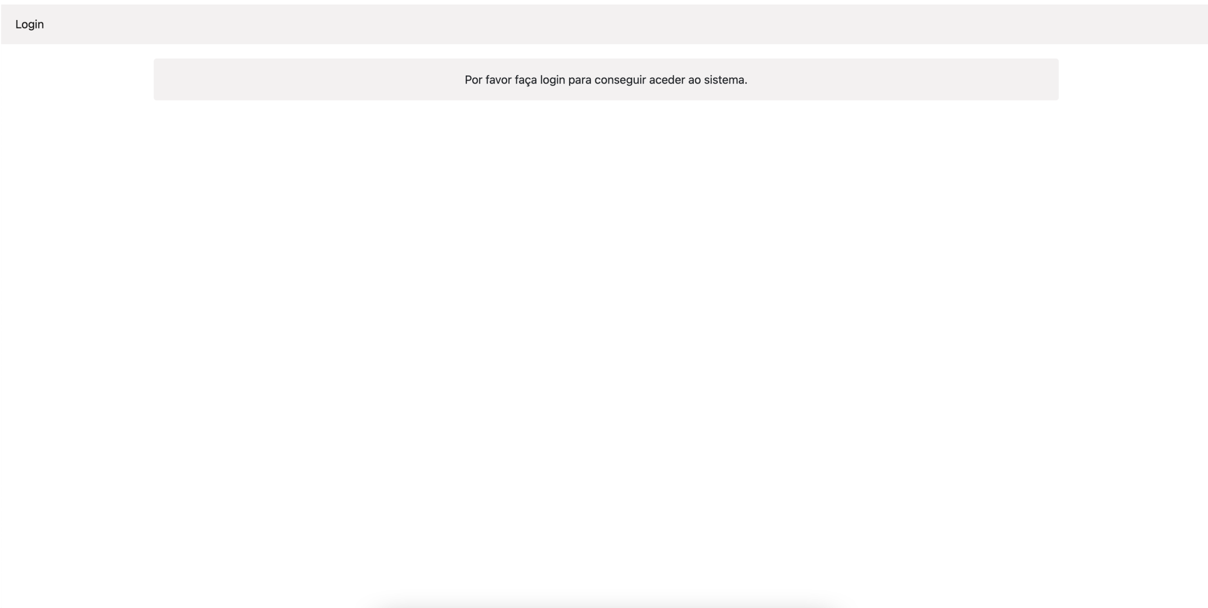


Figure A.1: First page to appear after we access the website. The students need to be logged in through the university website to be able to consult the exercises.

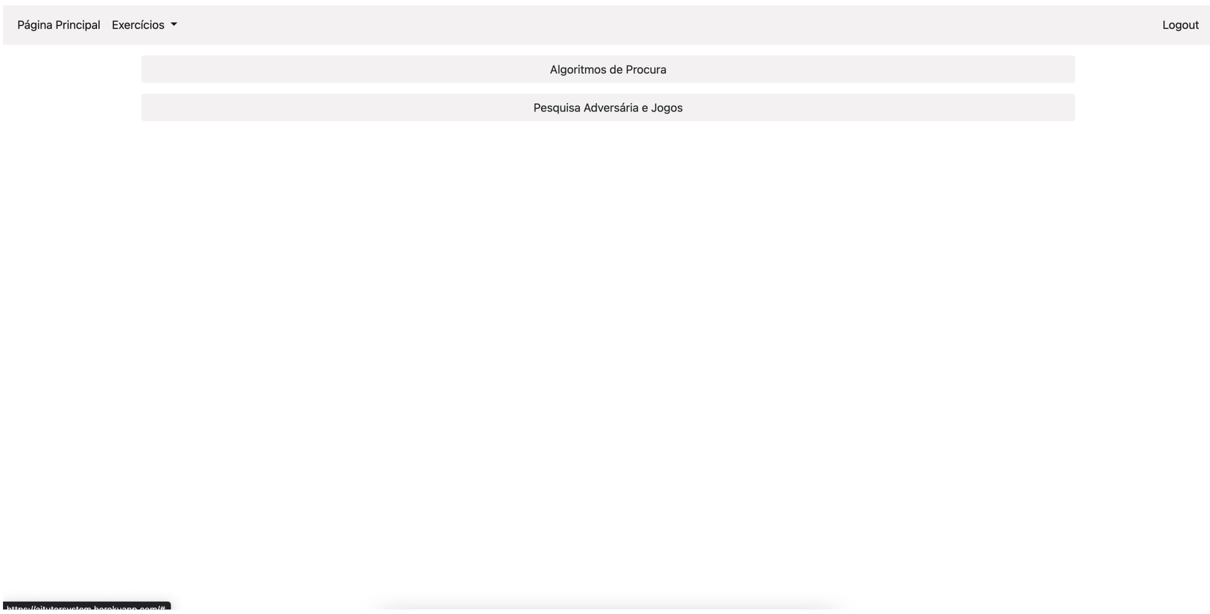


Figure A.2: Initial page after the login is completed. These page shows the type of exercises the students can solve.

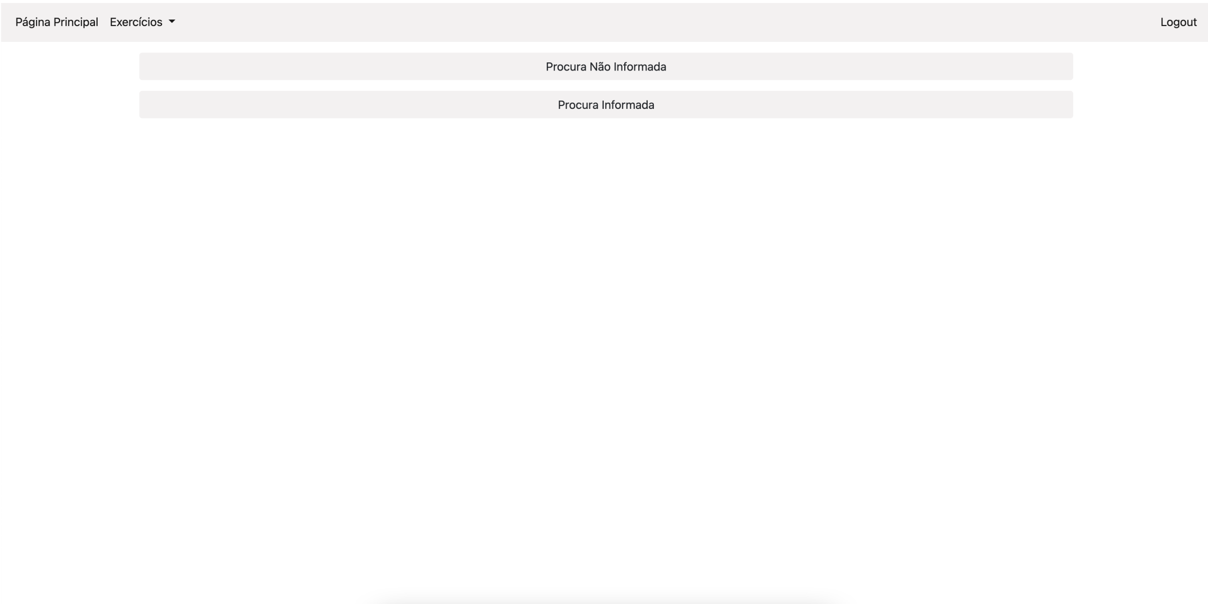


Figure A.3: After selecting exercises corresponding to search strategies, the student can choose what type of strategy they want to answer exercises from.

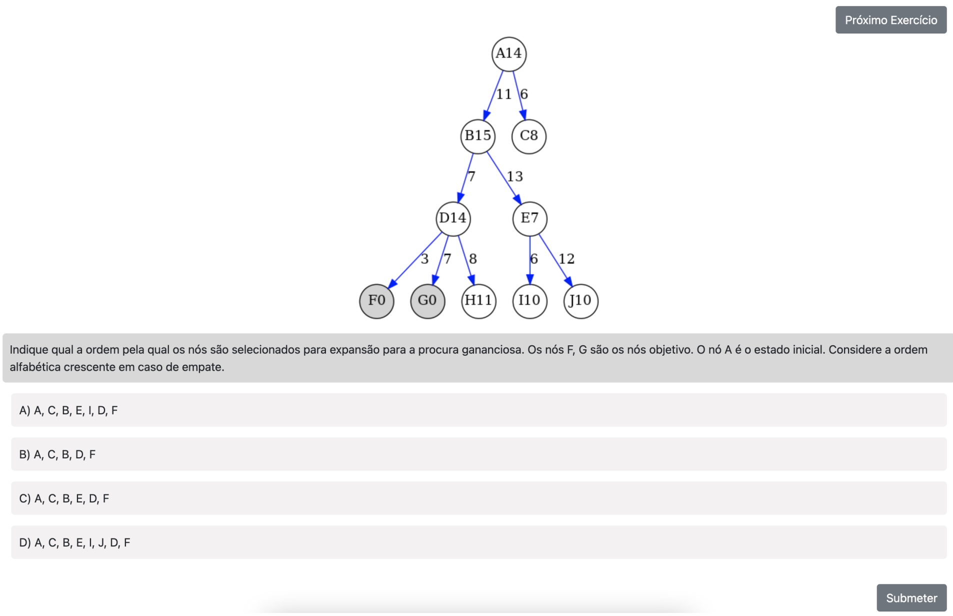


Figure A.4: After selecting the search strategy type, an exercise is presented. The student can select the button at the top and generate a different exercise, or they can select an option and submit their answer by selecting the button at the bottom of the page.

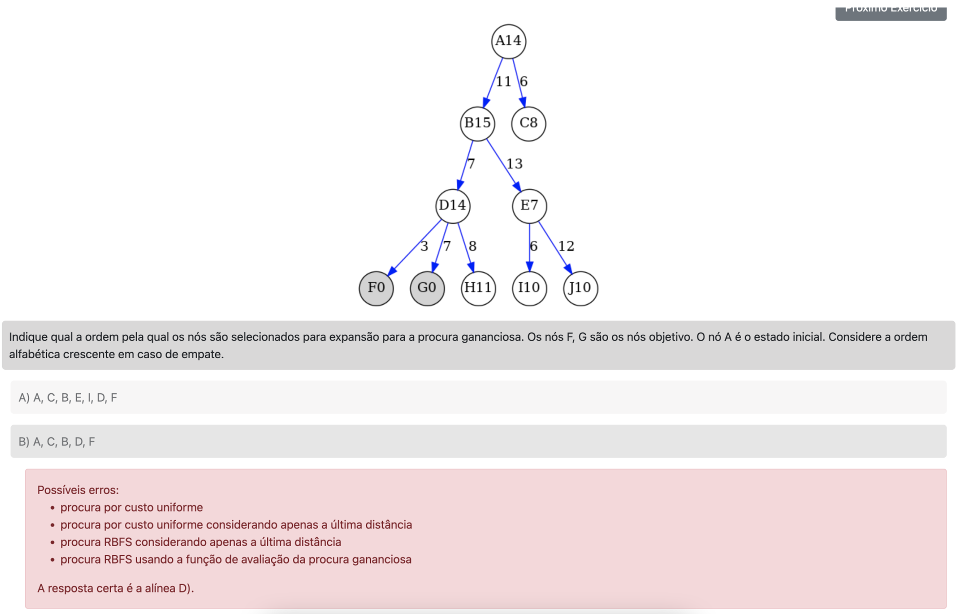


Figure A.5: After the student answers an exercise, the system indicates the true answer and possible mistakes made while solving the question.