Simulador de Testes Adaptativos Baseado na Proficiência do Aluno

Brandão, H. M. de L.
Pontifícia Universidade Católica de Goiás
Guedes, L. G. de R.
Universidade Federal de Goiás & Universidade Católica de Goiás

Resumo: A evolução da Internet e o uso desta transformou e continua transformando cada vez mais o contexto humano e, a educação, também está inserida nessa situação. Devido a isso, há ferramentas de apoio ao estudante, como sistemas avaliativos online. Apresentaremos nesse artigo um programa de simulação de uma avaliação online feita por um aluno com proeficiência determinada, a fim de coletar todos os dados a respeito do desempenho deste aluno, utilizando a teoria de Resposta ao Item. Verifica-se que ambos os métodos são subótimos para situações diferentes.

Palavras Chaves: Teoria de Resposta ao item, Testes Adaptativos Informatizados, Modelo Avaliativo.

1 Introdução

A Educação a Distância é uma tendência e um futuro da educação, contudo, ainda está em um processo de desenvolvimento e melhoria, por isso, se faz necessário o uso de tecnologias que ofereçam suporte à essa modalidade.

Nesse sentido, em um contexto de Educação a Distância preocupa-se com o método avaliativo, isto é, se este é eficiente e confiável. Portanto, convém simular tal sistema a fim de poder analisar empiricamente um método avaliativo a distância.

Dessa forma, a proposta deste artigo é elaborar e avaliar um programa cujo objetivo é simular uma avaliação (usando dois Modelos Avaliativos distintos) que siga a modalidade de Educação a Distância, a qual se fundamenta na Teoria de Resposta ao Item e da metodologia dos Testes Adaptativos Informatizados que servirão para selecionar questões e aferir o desempenho do aluno durante a avaliação. Espera-se que nas simulações a mediana dos níveis (de dificuldade) das questões feitas pelo aluno coincida com o nível de proeficiência desse.

A Teoria de Resposta ao Item (TRI) consiste em um modelo avaliativo cujo objetivo é avaliar o desempenho do aluno mediante a relação entre a probabilidade deste acertar um item a partir da habilidade do aluno, permitindo, assim, uma avaliação personalizada do aluno e não generalista como a adotada convencionalmente.

Por conseguinte, torna-se a avaliação mais congruente com o aluno posto que está irá analisar o quão difícil (probabilidade de acertar) a questão a ser feita pelo avaliado utilizando como parâmetro a habilidade deste e a dificuldade da questão, sendo esse a base do modelo usado pelo programa e proposto pelo matemático dinamarquês Georg Rasch (Baker & KIM, 2017):

$$P_i(\theta) = \frac{1}{1+\exp^{-1(\theta - b_i)}} \square$$

Em que:

- $P_i(\theta)$ é a probabilidade de um aluno com proficiência θ, responder a um item i corretamente;
- b_i é o índice de dificuldade do item.

Os valores de θ e $b_i$contemplam o domínio entre -2.0 e +2.0, com os níveis de dificuldade e habilidade crescentes.

2 Material e Métodos

Estudou-se dois modelos propostos para avaliação utilizando-se a Teoria de Resposta ao Item, os quais se diferem na maneira de selecionar o próximo item a ser feito pelo aluno e ambos propõem a ideia de tornar a avaliação intrínseca com o desempenho do aluno em questões de certo nível de dificuldade.

2.1 Modelo Avaliativo I

Neste modelo, há um banco de itens previamente cadastrado, no qual há questões de vários níveis de dificuldade, variando de zero (0) à dez (10). Sendo questões próximas a zero (0) consideradas fáceis e questões próximas a dez (10) consideradas difíceis, em que os níveis dessas questões são cadastrados inicialmente por um professor.

Inicialmente, o avaliado inicia fazendo um item de nível 5 e, quando ele acerta, a próxima questão terá 1 nível a mais de dificuldade enquanto ao errar a próxima questão diminui o nível de dificuldade em 1. Assim, o sistema sempre estará buscando desafiar o aluno até encontrar o nível de dificuldade que este está mais apto a fazer e, a partir disso, concluir sobre o rendimento do aluno na avaliação.

Por exemplo, um aluno inicia o teste com nível 5, ele acerta duas questões (a inicial e a do nível 6, disponibilizada após o acerto da de nível 5) consecutivas, dessa forma, será contemplado com uma questão de dificuldade 7 e caso ele erre, ele retornará a uma questão de nível 6.

Portanto, percebe-se que esse modelo busca a mediana do nível das questões feitas pelo aluno de modo incremental, devido a isso, independente a ordem em que o aluno acertou ‘x’ itens e errou ‘y’ o último nível dele será de: 5+(x-y), impossibilitando, assim, *outliers* ao se analisar somente o nível da última questão, permitindo, assim, utilizar o último nível feito pelo aluno para um outra análise (DAMANDO & GUEDES, 2005).

2.2 Modelo Avaliativo II

Neste modelo, assim como, no Modelo Associativo I este também há um conjunto de questões cadastradas com as dificuldades destas podendo variar da mesma forma como no primeiro modelo e a avaliação inicia no nível 5.

Não obstante, o processo de transição do nível de uma questão a outra é diferente, pois neste além de importar o nível da questão anterior importa se o aluno acertou ou não essa.

Desse modo, sempre que o aluno acertar uma questão independentemente se ele acertou ou não a anterior o nível da próxima questão será dado por:

$$n_{novo} = \frac{n_{atual} + n_{máx}}{2}$$

Caso o aluno responda incorretamente um questão mas acertou a anterior o nível da próxima questão será dado por:

$$n_{novo} = \frac{n_{atual} + n_{anterior}}{2}$$

Enquanto, se o aluno errar duas vezes consecutivas o nível do próximo questão será:

$$n_{novo} = \frac{n_{atual} + n_{mín}}{2}$$

Por exemplo, o aluno inicia a prova no nível 5 e acerta essa questão, logo, pela primeira regra, ele fará uma questão de nível 7.5 mas como trabalha-se com valores inteiros arredonda-se para cima (8), posteriormente, esse aluno erra a questão de nível 8, acarretando na primeira regra de erro, fazendo a média entre o nível atual (8) e da questão anterior (5) resultando em 6.5 mas quando o aluno erra arredonda-se para baixo, logo, (6) e se ele errar novamente será o caso da segunda regra de erro fazendo-se a média entra o nível atual (6) e o nível mínimo 0, logo, 3 e, consequentemente, 3.

2.3 Algoritmo

O algoritmo desenvolvido e todas suas versões apresentam a mesma estrutura lógica e interna apenas mudam nas funcionalidades implementadas e na otimização desses processos, sendo que todos foram implementados na linguagem C.

2.3.1 Versão 1

A primeira versão do programa disponibiliza uma simulação de duas provas (uma da Modalidade I e outro da Modalidade II) com a quantidade de questões determinadas em tempo de execução com o objetivo de testar a busca pelas questões do nível adequado a partir do cálculo do nível da questão a ser buscada correspondente a cada modalidade.

Assim, o programa apenas simula a prova adotando chance de 50% para qualquer questão independentemente da habilidade do aluno e do nível das questões, ou seja, o objetivo é simular se o programa busca corretamente as questões de nível apropriado.

O programa usa um sistema de busca aleatória, isto é, quando durante a avaliação, fosse calculado que a próxima questão deveria ser no nível x de dificuldade, começa a sortear entre as questões de nível x que não foram sorteadas uma para ser usada no teste. Entretanto, essa operação consume bastante tempo, posto que a função que gera número aleatório gera muitos números repetidos.

2.3.2 Versão 2

A segunda versão do programa tem como objetivo tornar a função de busca mais eficiente e implementar uma nova funcionalidade: calcular a mediana dos níveis feitos pelo aluno.

Desta forma, a fim de resolver o problema do tempo excessivo gastado pela função de busca de itens, implementou-se nessa função uma condição de buscar uma questão adequada (do nível correspondente e que não foi usada no neste) sequencialmente caso a função aleatória não encontrou uma nova questão não usada depois de N/4 vezes, sendo N o número de questões daquele nível.

Enquanto, a nova funcionalidade foi implementada mediante o armazenamento dos níveis das questões em tempo de execução e posteriormente ordena-se esses níveis mediante o método Bubble Sort (SAXENA, 2018), contudo esse também tornava o programa mais lento devido sua complexidade fazendo o tempo de execução se estender.

2.3.3 Versão Final

A versão final do programa teve como objetivo otimizar o tempo de execução e também implementar uma nova funcionalidade capaz de determinar a quantidade de questões de calibragem, isto é, a quantidade de questões que poderiam ser desconsideradas sem alterar a mediana. Lembrando que o princípio do programa é ter a mediana dos níveis semelhante ao nível de proeficiência do aluno, o qual é fornecido inicialmente no programa.

Assim, a fim de tornar a execução mais eficiente e gastar menos tempo foi trocado o método de ordenação do Bubble Sort (SAXENA, 2018) para o Quick Sort (ALI, 2018). Enquanto para tornar a avaliação congruente com o TRI a probabilidade do aluno acertar o item é dada pela habilidade (fornecida inicialmente na execução do programa) do aluno e o nível de dificuldade da questão como apresentado no tópico 2 (“Teoria de Resposta ao Item”). Sendo que a proeficiência o nível de dificuldade das questões são convertidos para o domínio destes na fórmula de Georg Rasch mediante o cálculo:

$$X = \left(\frac{x}{2.5} - 2 \right)$$

Em que:

- X é a proeficiência θ ou o nível da questão em um domínio de -2 a +2
- X é a proeficiência θ ou o nível da questão em um domínio de 0 a 10

O programa decide se o aluno acerta ou não as questões ao calcular a probabilidade do aluno acertar mediante a formula de Georg Rasch e pelo seguinte processo:

- Seja P(x) a chance de acertar uma questão, sorteia-se um número entre 0 e 1, se o número sorteado for maior que (1-P(x)) determina-se que o aluno acertou a questão caso contrário ele errou.

Por fim, a calibração foi feita obtendo a quantidade de questões iniciais que poderiam ser desconsideradas sem alterar a mediana dos níveis do aluno, contribuindo, assim, para uma avaliação que encontra o nível do aluno mais rapidamente.

Pseudocódigo

```
Main() {
TAM=Nº Questões;
Habilidade = Nº Proeficiência;
Mat[TAM][11] = AlocaMatriz();
vetModelo1[TAM] = AlocaVetor();
vetModelo2[TAM] = AlocaVetor();
mediaMediana1[TAM] = AlocaVetor();
mediaMediana2[TAM] = AlocaVetor();
mediaCalibragem1[TAM] = AlocaVetor();
mediaCalibragem2[TAM] = AlocaVetor()
vetModelo1 = modeloUm(mat,habilidade,TAM); // executa a modalidade I e retorna
um vetor com os níveis feitos
    vetAux1 = vetModelo1; // vetor auxiliar para salvar a sequência de níveis feitos.
   zera_controle_questões();
 vetModelo2 = modeloDois(mat,habilidade,TAM); executa a modalidade II e retorna um vetor com os
níveis feitos
vetAux2 = vetModelo1;
                ordena_vetor(vetModelo1); // ordena os níveis em ordem
        crescente
                ordena_vetor(vetModelo2);
                Mediana1 = calcular_Mediana(vetModelo1);
                Mediana2 = calcular_Mediana(vetModelo2);
                Calibragem1 = calcular_Calibragem(vetAux1);
                Calibragem2 = calcular_Calibragem(vetAux2);
                    mostrarResultados();}
int* modeloUm (mat[TAM][11],habilidade,TAM) {
    vet[TAM] = alocaVetor();
          nível = 5;
    contador = 0;
    para 0 até TAM {
        vet[] = nível; //armazena os níveis feitos
        enquanto(1){
        aleatório = sorteia()%TAM  // sorteia número entre 0 e TAM
contador++; // conta quantas vezes procurou uma questão = mat [aleatorio] [nível];
se(contador==TAM/4){
        enquanto(1){
questão=busca_sequencial(mat[][nivel]);  // busca questão na coluna do nível indicado
se(questão.controle==0) quebra;}}
se(questão.controle==0) quebra;}
        probAcerto=probabilidade_Acerto(habilidade,nivel);
        probErro=1-probAcerto;
        se (acerto<=probErro) acerto=0;  // errou a questão
        senão acerto=1;  // acertou a questão
        mat[aleatorio][nivel].controle=1;
        se(acerto){
```

```
                se(nivel<10) nivel++;
        senão{
                se(nivel>0) nivel--;}}
return vet;}
int* modeloDois (mat[TAM][11],habilidade,TAM) {
    vet[TAM] = alocaVetor();
          nível = 5;
    contador = 0;
    para 0 até TAM {
        vet[] = nível; //armazena os níveis feitos
        enquanto(1){
        aleatório = sorteia()%TAM  // sorteia número entre 0 e TAM
contador++; // conta quantas vezes procurou uma questão = mat [aleatorio] [nível];
se(contador==TAM/4){
        enquanto(1){
questão=busca_sequencial(mat[][nivel]);          // busca questão na coluna do nível indicado
se(questão.controle==0) quebra;}}
se(questão.controle==0) quebra;}
        probAcerto=probabilidade_Acerto(habilidade,nivel);
        probErro=1-probAcerto;
        se (acerto<=probErro) acerto=0;  // errou a questão
        senão acerto=1; // acertou a quetsão
        mat[aleatorio][nivel].controle=1;
                se(acerto){
                        se (nivel==9){
                                nivelAnterior=9;// guarda o nível anterior
                                anterior=1; //acertou a última questão
                                nivel=10;
                        }senão{
                                anterior=1;
                                nivelAnterior=nivel;
                                aux=(((float)nivel+10)/2);
                                nivel=ceil(aux)}}; // arredonda para cima
                senão{ // errou a questão
                        se (i==0){ // primeira questão
                        aux=(nivel+nivelAnterior)/2;
                        nivel=floor(aux); // arredonda para baixo
                        nivelAnterior=5;
                        }senão{
                                se (nivel==1){
                                        nivel=0;
                                        nivelAnterior=1;
                                }senão{
                                        se (anterior){// acertou a ultima
                                aux=((float)nivel+(float)nivelAnterior)/2;
                                nivelAnterior=nivel;
                                nivel=floor(aux);
                                }senão{
                                        nivelAnterior=nivel;
                                        aux=(float)nivel/2;
                                        nivel=floor(aux);
                                        }}}
                        anterior=0;}}
return vet;}
```

2.4 Resultados e Discussão

Os testes foram feitos simulando 20 provas consecutivas de cada habilidade((0), (3), (5), (6), (7), (8), (9) e (10)), para cada tamanho de prova com (5), (10), (15), (20), (25), (30), (35) e (40) questões, tornando-se possível obter algumas conclusões em relação as duas modalidades de prova. A seguir há 4 tabelas: Na Tabela 1 são apresentadas as médias das medianas obtidas quando um aluno de habilidade fornecida pela linha (da tabela) fez uma prova de tamanho dado pelas colunas sendo aplicado sobre estas a Modalidade I, na Tabela 2 são apresentadas as médias das calibragens obtidas quando um aluno de habilidade fornecida pela linha (da tabela) fez uma prova de tamanho dado pelas colunas (da tabela) sendo aplicado sobre estas a Modalidade I, na Tabela 3 são apresentadas as médias das medianas assim como a Tabela I mas agora aplicando a Modalidade II sobre as questões e na Tabela 4 são apresentadas as calibragens ,assim como a Tabela 2 mas neste teste aplicou-se a Modalidade II sobre as questões.

Tabela 1: Média das medianas pela Modalidade I

Habilidade	Tamanho da prova							
	5	10	15	20	25	30	35	40
0	3	2	1	1	1	1	1	1
3	4	3	5	3	3	2	3	2
5	3	3	4	4	5	4	4	4
6	5	5	6	4	5	5	5	5
7	5	5	6	6	6	6	8	6
8	5	6	6	7	6	7	6	7
9	6	7	7	7	8	7	6	7
10	6	6	7	8	8	8	8	8

Tabela 2: Média das calibragens pela Modalidade I

Habilidades	Tamanho da prova							
	5	10	15	20	25	30	35	40
0	1	0	1	1	2	2	2	3
3	1	2	0	1	3	2	3	2
5	1	1	1	1	2	2	3	2
6	1	0	2	1	2	2	3	3
7	1	0	1	1	3	2	5	2
8	1	1	3	1	2	1	3	3
9	1	0	1	1	1	2	3	3
10	1	0	2	1	2	1	3	3

Tabela 3: Média das medianas pela Modalidade II

Habilidades	Tamanho da prova							
	5	10	15	20	25	30	35	40
0	3	2	2	2	2	2	2	2
3	5	4	5	4	4	4	4	4
5	5	4	5	5	6	5	6	6
6	6	6	6	6	6	6	6	6

7	6	6	7	7	7	7	8	7
8	6	7	7	7	7	8	6	7
9	7	7	8	8	8	8	7	8
10	9	7	8	9	9	8	8	8

Tabela 4: Média das calibragens pela Modalidade II

Habilidades	Tamanho da prova							
	5	10	15	20	25	30	35	40
0	0	0	0	0	1	1	1	2
3	1	0	0	0	1	0	1	1
5	1	0	0	1	1	0	2	1
6	1	0	1	0	2	1	2	1
7	1	0	1	1	2	1	3	1
8	1	0	1	1	1	2	3	2
9	1	0	1	1	3	2	1	2
10	1	0	1	2	3	3	4	3

Para todos as simulações:

- A Modalidade II em 65% dos casos ofereceu ao aluno uma mediana de seus níveis maior que da Modalidade I, enquanto a Modalidade I teve 0% das vezes e em 35% tiveram a mesma mediana.
- A Modalidade II em 56% dos casos teve a mediana com valores mais próximos da proeficiência dada, enquanto a Modalidade II teve 15% dos casos com valores mais próximos e, consequentemente, 29% foram próximos da mesma forma.
- A Modalidade I teve 56% dos casos com calibragem maior que a da Modalidade II enquanto a Modalidade I conseguia uma maior calibragem em 9% dos casos e, consequentemente, a calibragem foi a mesma para 35% dos casos.

3 Conclusões

Destarte, observa-se que o programa consegue simular adequadamente ambas as Modalidades de prova e ser congruente com o proposto no TRI.

Dessa forma, é possível para um profissional da educação utilizar o programa para compreender como funcionaria uma certa condição de uma prova a ser aplicada a fim de averiguar se é o contexto mais adequado para aplicar tal avaliação. Assim como, descobrir o número ideal de questões de calibração para tornar a prova mais objetiva e eficaz.

Observa-se que a Modalidade II provém resultados maiores de mediana que a Modalidade I, por isso, mediante as simulações pode-se concluir que para avaliar os alunos com questões mais difíceis ou para ter uma prova mais congruente com um aluno de nível de habilidade alto convém utilizar a Modalidade II. Ademais, a Modalidade II também se mostrou mais precisa quanto aos valores das medianas serem próximos do nível de habilidade do aluno.

Enquanto, a Modalidade I provou-se mais adequada para provas mais curtas, pois ela necessita de menos questões para chegar na mediana a ser desenvolvida, visto que ela permite mais questões de calibração, ou seja, pode-se retirar mais questões da avaliação sem prejudicar a mediana dos níveis das questões.

Referências

ALI, I. et al. **Performance Comparison between Merge and Quick Sort Algorithms in Data Structure**. International Journal of Advanced Computer Science and Applications, v. 9, n. 11, p. 192-195, 2018.

BAKER, F. B. **The Basis of Item Response Theory**. Second Edition. ERIC Clearinghouse on Assessment and Evaluation, 2001.

BAKER, F. B.; KIM, S. **The basics of item response theory using R**. New York, NY: Springer, 2017

SANTOS, F. D.; GUEDES, L. G. de R. **Testes Adaptativos Informatizados baseados em teoria de resposta ao item utilizados em ambientes virtuais de aprendizagem**. RENOTE, 2005, vol. 3, no 2.

SAXENA, Shweta. QB Sort: **An Efficient computational Algorithm for advance networks**. In: 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU). IEEE, 2018. p. 1-4.