Avaliação do sistema de recomendação do Beecrowd

Carlos Amorim[1], **Raoni Ferreira**[1], **Catarina Costa**[1]

[1]Centro de Ciências Exatas e Tecnológicas - Universidade Federal do Acre (UFAC)
Rio Branco – AC – Brasil

carlos.amorim@sou.ufac.br, raoni.ferreira@ufac.br,

catarina.costa@ufac.br

Abstract. *This study explores the use of a recommendation module for programming activities on an online judge platform, aiming to personalize learning and facilitate student progress tracking. An experimental methodology was implemented in which two groups were assigned programming exercise lists: one group with activities directly selected by the instructor and the other with activities recommended by the system. The lists were adjusted according to the characteristics and needs of each group. The results showed that, while the recommendation module offers an automated approach, the instructor-customized activities proved more effective in some aspects, such as completion rate and average score.*

Resumo. *Este trabalho explora o uso de um módulo de recomendação de atividades de programação em uma plataforma de juízes online, com o objetivo de personalizar o aprendizado e facilitar o acompanhamento do progresso dos alunos. Para isso, foi implementada uma metodologia experimental em que duas turmas foram submetidas a listas de exercícios de programação, uma delas com atividades selecionadas diretamente pelo professor e outra com atividades recomendadas pelo sistema. As listas foram ajustadas com base nas características e necessidades de cada turma. Os resultados mostraram que, embora o módulo de recomendação ofereça uma abordagem automatizada, as atividades personalizadas pelo professor mostraram-se mais eficazes em alguns aspectos, como taxa de conclusão e pontuação média.*

1. Introdução

As disciplinas de programação são essenciais no processo formativo dos discentes de diversos cursos da área de Computação [Zorzo et al. 2017], por ser o passo inicial para o desenvolvimento de habilidades como as de abstrair e resolver problemas, raciocinar e pensar logicamente [Raabe 2005]. Apesar desta importância, o processo de aprendizagem em programação é um obstáculo para alunos iniciantes, sendo causa dos altos índices de reprovação e evasão ao final de cada semestre [Pfitscher et al. 2023].

Cada vez mais, professores de programação têm adotado plataformas de Juízes Online (JOs) nos planos de ensino dessas disciplinas com objetivo de apoiar o processo de aprendizagem em programação [Pessoa et al. 2021, Cruz et al. 2022]. Plataformas de JOs geralmente mantêm um grande conjunto de problemas (banco de questões) organizados por tópicos (ou categorias) e por nível de dificuldade. Adicionalmente, tais sistemas

fornecem uma avaliação automática e instantânea das soluções submetidas pelos alunos da plataforma [Zhao et al. 2018].

Originalmente, JOs foram projetados para aprendizagem autodirigida de programação, sem ajuda de um instrutor ou interação com instrutor [Zhao et al. 2018]. No entanto, professores têm empregado JOs para tornar o ensino de programação mais prático e interativo. Alguns trabalhos apontam uma melhora no rendimento das turmas de Algoritmos quando JOs são combinados com metodologias ativas [Pessoa et al. 2021, Cruz et al. 2022]. Assim, professores utilizam JOs durante a preparação de listas de exercícios, organizando os problemas com base no nível de dificuldade do exercício, com base nos conceitos abordados [Zhao et al. 2018], ou com base nas dificuldades individuais e coletivas monitoradas pelo próprio professor [Ramos et al. 2023]. Dessa forma, alunos podem ser incentivados a praticarem os conceitos através da escrita de códigos que solucionem um problema na plataforma JOs, podendo assim tirar dúvidas na aula. Logo, espera-se que esses alunos, ao longo das aulas, tenham uma evolução das habilidades de programação adquiridas [Pessoa et al. 2021, Cruz et al. 2022].

O aumento contínuo de contribuições de novos exercícios para o banco de questões motivou mantenedores dessas plataformas e pesquisadores a desenvolverem um módulo que implementa um sistema de recomendação de problemas para apoiar o trabalho do professor e o aprendizado do aluno. Alguns trabalhos na literatura, a exemplo de Júnior et al. [2020] , tem mostrado benefícios dos sistemas de recomendação para consolidação do conhecimento de programação dos alunos. Tal estudo indica que sistemas de recomendações implementados em JOs podem ser usados para sugerir problemas parecidos, em termos de nível da questão e conceito de programação exigido, com um problema alvo resolvido anteriormente pelo aluno [Júnior et al. 2020]. No entanto, a partir desta constatação, surge o seguinte questionamento – *o uso de recomendações de exercícios para turmas iniciantes, com objetivo de exercitar dificuldades coletivas em torno de conceitos de programação que foram previamente identificadas, contribuem para melhoria do desempenho dessas turmas?*

A nossa hipótese é que tais sugestões retornadas pelo recomendador podem beneficiar alunos e professores. Do ponto de vista do aluno, o recomendador possibilita que esses alunos identifiquem suas próprias lacunas no conhecimento e os incentivem a buscar resolver variantes do problema no JOs. Por exemplo, se os alunos estão com dificuldades com geometria, o sistema pode recomendar problemas mais simples antes de avançar para os mais complexos. Por outro lado, se os alunos já dominam problemas de *strings* ou já demonstram muito interesse, podem receber problemas semelhantes e mais complexos, evitando que percam tempo com exercícios triviais. Do ponto de vista do professor, o sistema pode recomendar problemas com base no conteúdo da disciplina, com base nas dificuldades percebidas previamente pelo docente ou com base em métricas de monitoramento usadas sobre as listas e possam sugerir dificuldades individuais ou coletivas (por exemplo, números de tentativas, taxas de acertos, frequências de erros em determinados conceitos, etc). Nestes casos onde essas deficiências são notadas, o professor poderia intervir, por exemplo, após a atividade, revisar o conteúdo das listas e ajustar o sistema para que sugira novos problemas semelhantes e que reforcem o conteúdo revisado.

Assim, este artigo tem como objetivo realizar um comparativo das recomendações de exercícios de programação para turmas iniciantes em ambiente de Juízes Online.

Para tanto, realizamos um estudo de caso em um curso de extensão de programação Python para iniciantes utilizando a plataforma Beecrowd, onde detalhamos o impacto das recomendações de listas de exercícios sugeridas automaticamente pelo módulo de recomendação implementado no Beecrowd e das recomendação que foram sugeridas a partir de uma seleção feita pelo professor. Em ambas as listas de recomendações, os problemas foram extraídos do banco de questões disponibilizado no Beecrowd[1].

Este artigo está organizado em cinco seções. A Seção 2 descreve os trabalhos relacionados. A Seção 3 detalha as etapas da metodologia empregada para o estudo de caso. A Seção 4 apresenta os resultados encontrados e uma discussão em torno deles. E por fim, a Seção 5 descreve as conclusões, lições aprendidas e as possibilidades de trabalhos futuros.

2. Trabalhos Relacionados

A tarefa de recomendação de listas de problemas para turmas de programação é tratada de diferentes formas na literatura. O trabalho de Sanches et al. [2023] apresenta um estudo que propôs um sistema de recomendação utilizando dados do Beecrowd, considerando as submissões históricas. Um estudo de caso foi realizado com 54 estudantes da disciplinas de programação. Os estudantes submeteram suas soluções aos problemas da plataforma Beecrowd ao longo de 30 dias. Os dados foram tabulados e o modelo proposto foi aplicado para análise. O sistema buscou explorar o conhecimento coletivo gerado pelas interações e soluções apresentadas pelos estudantes. Os autores identificaram alguns problemas na definição do sistema por parte dos alunos, como a falta de clareza sobre seus próprios conhecimentos, dificuldades na resolução de problemas semelhantes e uma seleção limitada e pouco diversificada de problemas.

O trabalho de Ferreira et al. [2022] apresenta um modelo para avaliação de problemas de programação e um dos resultados do trabalho é a proposição de um sistema de recomendação utilizando a base de dados do Beecrowd. No estudo os autores realizaram uma simulação de recomendações de problemas com base nas habilidades dos alunos. A simulações das recomendações foram feitas a partir do 31º envio de cada usuário, para calibrar os valores de habilidade do aluno e as dificuldades do problema. O modelo estimou a maneira possível de resolver os problemas levando em consideração as habilidades individuais de cada usuário envolvido na solução do problema.

O trabalho de Júnior et al. [2020] propõe e avalia métodos para recomendação automática de problemas em juízes online, em que as recomendações são realizadas a partir de um problema alvo, resolvido anteriormente pelo aluno. Os dados foram coletados a partir do JO CodeBench, desenvolvido na Universidade Federal do Amazonas (UFAM). O sistema de recomendação adota Processamento de Linguagem Natural e utiliza o conceito de similaridade para realizar a recomendação.

Diferente dos trabalhos anteriores apresentados, nosso foco foi realizar um comparativo entre duas estratégias de recomendação de lista de exercícios tipicamente usados para auxiliar no aprendizado de programação em ambiente de Juízes Online: lista de exercícios sugerida pelo professor e a lista sugerida pelo Beecrowd. Ambas as recomendações foram geradas a partir das informações extraídas do diagnóstico aplicado para as turmas participantes do estudo.

[1]https://judge.beecrowd.com/

3. Etapas do Estudo

3.1. Organização das turmas participantes

O estudo foi realizado durante a oferta do curso de extensão de Introdução à Programação com Python, oferecido a alunos de graduação ingressantes ou com dificuldade em programação em áreas de tecnologia da Universidade Federal do Acre. Os 32 participantes do curso foram divididos aleatoriamente em duas turmas. Essas turmas receberam o mesmo conteúdo programático e realizaram as mesmas práticas em laboratório. No entanto, ao fim de cada aula, tais turmas foram avaliadas com base em dois tipos distintos de recomendações de exercícios. A turma que recebeu exercícios selecionados pelo professor foi denominada TurmaRecProf, enquanto que a turma que recebeu exercícios sugeridos pelo módulo de recomendação foi chamada de TurmaRecBee. Os alunos não foram informados sobre essa diferença de atribuição de exercícios.

3.2. Questionário diagnóstico

Antes do início do curso, foi aplicado um questionário diagnóstico para ambas as turmas com o objetivo de identificar o perfil dos alunos, o conhecimento e sua experiência prévia em programação. Além disso, identificar dificuldades em fazer o uso de conceitos básicos de programação para resolver problemas. Uma listagem das perguntas feitas para as turmas participantes é descrita a seguir.

1. Você está cursando (ou já cursou) a disciplina de Algoritmos?
    (a) Sim.
    (b) Não.
2. Se você já cursou, foi aprovado?
    (a) Sim.
    (b) Não.
3. Qual nível de conhecimento em programação você julga estar?
    (a) Abaixo do básico.
    (b) Básico.
    (c) Médio.
    (d) Avançado.
4. Qual nível de dificuldade para aprender programação você julga sentir?
    (a) Pouca dificuldade.
    (b) Dificuldade moderada.
    (c) Muita dificuldade.
5. Qual desse(s) tópico(s) de programação que você sente mais dificuldade?
    (a) Funções
    (b) Laços de repetição
    (c) Operadores lógicos
    (d) Estruturas condicionais
    (e) Tipagem de dados
6. Quais dessa(s) categoria(s) de problema(s) você sente mais dificuldade?
    (a) Strings
    (b) Matemáticas
    (c) Geometria
    (d) Paradigmas

As informações coletadas e estatísticas extraídas do questionário diagnóstico permitiram que o professor do curso decidisse quais os exercícios do repositório de problemas iria selecionar para a sua lista de exercícios e ajustasse os parâmetros de entrada do módulo de recomendação para geração automática da lista de exercícios.

3.3. Planejamento das Aulas e Seleção de Questões

O curso foi organizado em aulas presenciais duas vezes por semana, com sete encontros, totalizando 20 horas aulas, com atividades práticas em cada aula. Os conteúdos trabalhados foram: Função *print*, Função *input*, Operadores Aritméticos, Tipos primitivos, Variáveis, Conversão de tipos, Concatenação de *strings*, Interpolação de *strings*, Procedência de operadores aritméticos, Formatação de strings, Condicionais (if, else), Operadores lógicos, Ternário, Laços de repetição, Vetores e Matrizes, Funções com string (split, replace,...) e Tratamento de exceções.

As questões foram selecionadas com base no diagnóstico inicial, que apontou as principais dificuldades dos alunos em funções (59,3%), laços de repetição (37%) e estruturas condicionais (25,9%), além de temas como matemática (33,3%) e strings (22,2%). Para a TurmaRecProf, o professor selecionou manualmente questões do Beecrowd, priorizando problemas que exemplificassem os conceitos abordados em aula, com uma progressão didática desde exercícios básicos (ID 1000) até desafios que integravam múltiplos conceitos (ID 1020). Já para a TurmaRecBee, o sistema de recomendação foi configurado para sugerir automaticamente duas questões por semana, considerando apenas problemas de nível 1 e 2 nas categorias Iniciante, Ad-Hoc, Strings e Matemáticas, sem filtros adicionais.

3.4. Configuração das Recomendações de Exercícios

Ambas as turmas receberam sete questões, começando com uma atividade comum (ID 1000) para familiarização com a plataforma. A TurmaRecProf teve exercícios selecionados para reforçar conceitos específicos, como operações matemáticas (ID 1005) e problemas contextualizados (ID 1020). A TurmaRecBee, por sua vez, seguiu recomendações automáticas baseadas no desempenho médio de usuários, incluindo questões mais aplicadas (ID 1017) e introdução a paradigmas básicos (ID 2861). O objetivo foi comparar a eficácia das recomendações manuais (focadas no diagnóstico da turma) com as automáticas (baseadas em dados agregados da plataforma).

A especificação dos problemas geralmente segue um padrão: entrada e saída de dados sugerida. Em alguns problemas, a especificação é acompanhada de casos de testes, o que torna possível identificar tanto o tipo de dado quanto se o problema faz uso de uma estrutura, como repetição, vetor, matriz, etc. Dessa forma é possível classificá-los quanto ao conhecimento necessário para a realização e qual estrutura (caso haja) predomina no problema.

Decidimos por experimentar duas estratégias de recomendação, uma para TurmaRecBee e outra para TurmaRecProf, conforme descrito anteriormente na seção 3.1. Para ambas, procuramos recomendar exercícios, baseados no conteúdo das aulas, categoria dos problemas e nível esperado do aluno para resolvê-los. Limitamos apresentar um total de sete exercícios, que são exibidos gradativamente aos alunos, ao invés de apresentá-los todos de uma única vez. Assim, quando o professor encerrava uma aula, logo em seguida era disponibilizado o exercício para o aluno desenvolvê-lo individualmente.

A recomendação baseada na seleção do professor (TurmaRecProf) foi resultante da seleção manual do docente de quais exercícios do repositório de problemas do Beecrowd o docente julga adequado fazer parte da lista. Enquanto que o módulo de recomendação (TurmaRecBee) foi resultante da definição, por parte do docente, dos

parâmetros disponíveis no módulo para criação de uma lista. Até o momento, os parâmetros a serem configurados disponibilizados pelo Beecrowd são: categoria do problema, o nível do exercício e uma entrada textual onde pode ser inserida uma parte do problema.

Os repositórios de exercícios do Beecrowd são classificados geralmente em 7 categorias de problemas: iniciante, ad-hoc, strings, estruturas e bibliotecas, matemáticas, paradigmas e grafos. Essas classes de exercícios são tipicamente usadas em competições de programação. Como não é o objetivo deste artigo avaliar cenários de competições, decidimos por selecionar um subconjunto de classes: iniciante, ad-hoc, strings e matemáticas. Tal subconjunto representa questões para quem ainda está se habituando aos padrões de exercícios vistos em competição, que podem ser praticados em um curso direcionado para iniciantes, e que exploram aspectos como a contextualização, o funcionamento da entrada e saída de dados, e exemplos de casos de testes na especificação do problema.

A classe **Iniciante** é composta por exercícios extremamente fáceis, que requerem apenas conhecimento dos tipos de dados, estruturas de seleção/repetição e operações matemáticas básicas. **Ad-hoc** são problemas muito específicos, cada um com uma resolução específica, cujo o grau de dificuldade pode variar muito. Tais problemas modelam situações do mundo real. Um típico problema Ad-hoc no repositório é desenvolver uma solução que conte o número de vezes que um jogador ganhou cara ou coroa, dado uma sequência numérica de resultados das partidas entre dois jogadores. Problemas Ad-hoc demandam um entendimento do problema específico para elaborar uma solução que pode ser resolvidos com estruturas de seleção/repetição, operações lógicas e aritméticas.

A classe **Matemáticas** normalmente apresenta enunciado bem contextualizado para facilitar (ou dificultar, em alguns casos) a resolução. Os exercícios exploram conceitos matemáticos, fórmulas e lógica matemática no algoritmo. Se assemelham ao ad-hoc por possuir grande variação na dificuldade, tal como, desde operações aritméticas simples até a utilização de integrais. Por fim, ***Strings*** são problemas que trabalham, em essência, com a manipulação de caracteres e palavras, sendo necessário desenvolver algoritmos para os mais diversificados propósitos. A diferença de resolução desses problemas pode variar significativamente dependendo da linguagem de programação utilizada, por conta das funções existentes dentro de cada linguagem.

3.5. Análises do questionário diagnóstico

O questionário foi respondido por 27 alunos. A seguir são apresentados as principais estatísticas extraídas e suas respectivas análises.

1. **Alunos que já cursaram ou estão cursando a disciplina de Algoritmos, bem como aqueles que foram aprovados:** A análise revelou que 81,5% dos alunos já havia cursado e sido aprovado, enquanto 18,5% ainda não obtiveram aprovação.
2. **A autopercepção do nível de conhecimento em programação:** Cerca de 48,1% afirmaram não ter o básico de conhecimento, enquanto que 40,7% afirmaram ter conhecimento básico e 11,1% julgaram estar em um nível médio de conhecimento.
3. **A autopercepção do nível de dificuldade para aprender programação:** O sentimento de dificuldade no geral variou de muito difícil a dificuldade moderada (cada uma, correspondeu a 44,4%), pouca dificuldade (7,4%) e nenhuma dificuldade (3,8%).

4. **Tópicos de programação mais difíceis:** Os tópicos que foram relatados como mais difíceis foram funções (59,3%), laços de repetição (37%) e estruturas condicionais (25,9%).
5. **Tópicos de programação menos difíceis:** Os tópicos que foram relatados como menos difíceis incluem operadores lógicos (59,3%), laços de repetição (33,3%) e tipos de dados (33,3%).
6. **Categorias de problemas vistos na disciplina de Algoritmos que sentem mais dificuldades:** As categorias que mais foram citadas são relacionadas a problemas que aplicam conceitos matemáticos (33,3%), seguidos por geometria (29,6%) e manipulação de *strings* (22,2%).

De maneira geral esses números apontam que o público alvo do experimento é formado por alunos que em sua maioria foram aprovados na disciplina de Algoritmos, mas não se sentem seguros com os conhecimentos obtidos em programação para avançar nos seus estudos de algoritmos. Foi verificado que cerca de 88,9% julga ter conhecimento variando de não ter o básico ao básico em programação. Esse percentual foi similar quando notamos que a dificuldade em aprender programação variou de difícil a dificuldade moderada (88,8%).

Em relação aos tópicos de programação, as maiores dificuldades são relacionadas a funções, aos laços de repetição e as estruturas de condicionais, que de fato são conceitos básicos na programação de computadores. Essas dificuldades são reforçadas quando os estudantes afirmam problemas com questões que exploram conceitos matemáticos, de geometria e de manipulação de *strings*, que são categorias de questões que representam vários tipos de problemas fundamentais encontrados em computação.

Por fim, ao considerarmos essas análises do questionário diagnóstico, definimos que para além da adequação do conteúdo das aulas, as turmas receberiam exercícios compatíveis com o nível de conhecimento que se espera dos alunos iniciantes.

3.6. Métricas usadas para medir as recomendações

As métricas usadas para medir o o desempenho das turmas sobre os exercícios recomendados foram as seguintes:

- **Resolvidos:** Número de vezes que o exercício foi resolvido corretamente pelos alunos.
- **Tentativas:** Número de vezes que o exercício foi tentado, ao menos uma vez, independente de ter sido resolvido corretamente ou não.
- **Média de Tentativas:** Média de tentativas realizadas para resolver o exercício.
- **Média de Tentativas até resolver:** Média de tentativas realizadas até solucionar o exercício.
- **Tempo médio:** Média de tempo (em minutos) levado para resolver os exercícios.

4. Resultados das Recomendações

Com a divisão em duas turmas, ficaram 16 participantes em cada turma, finalizando 14 e 13 nas turmas TurmaRecProf e TurmaRecBee, respectivamente. A Figura 1 apresenta uma imagem do encerramento com as duas turmas participantes e os instrutores.

Figura 1. Encerramento do curso.

Tabela 1. Desempenho das turmas após receberem as recomendações de exercícios.

Turma	Média de exercícios	Média de tentativas (resolvidas)	Média de tentativas (não resolvidas)	Média de Tempo (minutos)
TurmaRecProf	5,18	5	0,19	46,05
TurmaRecBee	4,73	4,53	0,2	55,22

A Tabela 1 apresenta os valores de desempenho obtidos para as duas turmas após aplicação das recomendações. Logo, os valores desta tabela correspondem a média sobre esse quantitativo de alunos que ficaram até o fim do curso.

Os valores de média de cada turma são bem próximos. Há uma ligeira superioridade dos valores para a turma que recebeu recomendações do professor. Ao verificar a média de exercícios da lista realizados pela TurmaRecProf foi um pouco maior que cinco questões e na TurmaRecBee foi de 4,73. Ao considerar a média de tentativas (resolvidas), verifica-se que a TurmaRecBee resolveu menos de cinco exercícios, enquanto que a TurmaRecProf resolveu em média cinco.

Ao verificar a média de tentativas (não resolvidas), que mede quantas questões da lista foram tentadas mas não resolvidas pela turma, percebe-se que em geral as tentativas de resolução em ambas as turmas foi similar e quase sempre encontravam a solução dos exercícios. Ao comparar as médias de tempo para resolver as sete questões, a TurmaRecProf mostrou respondê-las em menos tempo, em torno de 46 minutos em média (comparado aos 55 minutos da TurmaRecBee). Isso pode sugerir que a abordagem automática recomendou questões mais desafiadoras do que a abordagem manual.

As informações das Tabelas 2 e 3 permitem visualizar melhor o desempenho em cada um dos sete exercícios recomendados para cada turma. Para facilitar as análise das tabelas, procuramos identificar a categoria e o nível do exercício. Note que as questões podem pertencer a mais de uma categoria de problema e, dentro de uma mesma classe de problema, pode haver diferentes níveis de dificuldades de exercícios. A questão de id 1000 é extremamente fácil (nível 1 de dificuldade) e apareceu em ambas as listas com

objetivo dos alunos terem o primeiro contato com o padrão de especificação tipicamente encontrados nos repositórios de problemas. Exercícios da classe Ad-hoc, como 1020 e 1142 na Tabela 2, podem apresentar diferentes níveis de dificuldades.

Tabela 2. Desempenho em cada exercício recomendado para TurmaRecProf

id	Categoria	Nível do exercício	Resolvidos	Tentativas	Média de tentativas	Média de tentativas até resolver
1000	Iniciante	1	14	25	1.8	1.8
1003	Iniciante	1	13	55	3.7	3.4
1005	Iniciante, Matemáticas	2	13	89	5.7	6.0
1020	Iniciante, Ad-Hoc	2	12	33	2.8	2.8
1078	Iniciante, Matemáticas	1	10	27	2.7	2.7
1142	Iniciante, Ad-Hoc	1	9	14	1.6	1.6
2483	Ad-Hoc, Strings	1	9	17	1.9	1.9

Tabela 3. Desempenho em cada exercício recomendado para TurmaRecBee

id	Categoria	Nível do exercício	Resolvidos	Tentativas	Média de tentativas	Média de tentativas até resolver
1000	Iniciante	1	13	19	1.5	1.5
1038	Iniciante	1	10	67	5.2	4.7
1004	Iniciante, Matemáticas	1	9	30	3.3	3.3
1017	Iniciante, Matemáticas	1	9	12	1.3	1.3
1073	Paradigmas, Matemáticas	1	8	13	1.6	1.6
2861	Paradigmas	1	12	16	1.3	1.3
3046	Paradigmas, Matemáticas	1	7	9	1.3	1.3

Ao fazer um comparativo de desempenho entre as turmas, a média de tentativas por exercício da TurmaRecProf (Tabela 2) foi ligeiramente maior do aquelas obtidas pela TurmaRecBee (Tabela 3). Questões da classe matemática e ad-hoc (ids 1004, 1005, 1020 e 1078), que costumam ter problemas que exigem soluções mais elaboradas, por exemplo, foram as que apresentaram maiores médias. Curiosamente, exercícios de categoria iniciante (ids 1003 e 1038) apresentaram médias de tentativa elevada também, o que sugere exercícios extremamente fáceis (ou seja, nível 1 de dificuldade) podem facilmente levar a erros se a resposta não for devidamente revisada antes de ser submetida.

Outro destaque é a facilidade que a TurmaRecBee (Tabela 3) teve para resolver corretamente os exercícios recomendados em comparação com a TurmaRecProf (Tabela 2). Pode haver múltiplas possibilidades por traz desse comportamento. A maioria dos exercícios recomendados pelo professor para a TurmaRecProf precisou, em média, ser tentados mais vezes, entre aqueles alunos que não desistiram das questões, até conseguir respondê-las corretamente. Apesar da recomendação do professor apresentar exercícios de nível 2, isso não foi determinante para que a lista do professor fosse mais difícil no geral. Exercícios de nível 1, considerados extremamente fáceis, mas que podem exigir cautela na elaboração da solução e revisões do código fonte antes de serem submetidos, também contribuíram para elevar essas médias na abordagem manual. Neste sentido, apesar de ter levado mais tempo para conclusão da lista (Tabela 1), a TurmaRecBee precisou de menos tentativas até conseguir, e portanto a abordagem automática impactou positivamente na pontuação média nesta turma em particular.

5. Conclusão

A comparação entre as turmas TurmaRecProf e TurmaRecBee revelou que a abordagem tradicional do professor foi ligeiramente mais eficaz. A TurmaRecProf alcançou uma taxa de conclusão de 78,57% e uma pontuação média de 550 pontos, enquanto a TurmaRecBee obteve 65,93% de conclusão e 461,53 pontos em média.

A análise das tentativas de resolução mostrou que a TurmaRecProf teve uma média de tentativas por questão entre 1,6 e 2,8, indicando uma eficiência razoável. Já a TurmaRecBee apresentou uma variação maior, com algumas questões exigindo até 5,2 tentativas, o que sugere que as atividades recomendadas pelo sistema podem ter sido mais desafiadoras ou menos alinhadas às habilidades dos alunos. Esses resultados destacam que, embora o sistema de recomendação do Beecrowd tenha potencial para personalizar o aprendizado, ele ainda precisa de ajustes para se alinhar melhor às necessidades individuais dos alunos.

O estudo apresentou algumas limitações, como a generalização do sistema de recomendação, que se baseia no desempenho geral dos usuários da plataforma e não no contexto específico das turmas. Além disso, fatores externos, como motivação individual e tempo disponível para estudo, podem ter influenciado os resultados. O período limitado de coleta de dados também restringiu a análise do impacto a longo prazo do sistema de recomendação.

Como trabalhos futuros, podem ser levantadas algumas possibilidades: Comparação com outras plataformas através de uma investigação mais abrangente que compare o sistema de recomendação do Beecrowd com outras plataformas de ensino de programação; e, Exploração de técnicas de recomendação utilizando técnicas mais avançadas de IA, como aprendizado profundo ou análise de desempenho anterior mais detalhada, e assim oferecer recomendações ainda mais precisas e personalizadas.

Referências

Cruz, A., Neto, C. S., Cruz, P., and Teixeira, M. (2022). Utilização da plataforma beecrowd de maratona de programação como estratégia para o ensino de algoritmos. In *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 754–764, Porto Alegre, RS, Brasil. SBC.

Ferreira, F. Z., de Souza, R. L., Vargas, A. P., de Lemos Teixeira, D., dos Santos, M. N., de Oliveira Paes, W., dos Santos, R. A. P., Tonin, N., de Oliveira Evald, P. J. D., and da Costa Botelho, S. S. (2022). Model for evaluation of multiple abilities programming problems in online massive environments. *Journal of the Brazilian Computer Society*, 28(1):104–117.

Júnior, H., Pereira, F., Oliveira, E., Oliveira, D., and Carvalho, L. (2020). Recomendação automática de problemas em juízes online usando processamento de linguagem natural e análise dirigida aos dados. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, pages 1152–1161, Porto Alegre, RS, Brasil. SBC.

Pessoa, M., Melo, R., Haydar, G., Oliveira, D., Carvalho, L., Oliveira, E., Conte, T., Pereira, F., Rodrigues, L., and Isotani, S. (2021). Uma análise dos tipos de jogadores em uma plataforma de gamificação incorporada a um sistema juiz on-line. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 474–486, Porto Alegre, RS, Brasil. SBC.

Pfitscher, R., Camargo, L., Moreira, B., Wang, C., Zedral, R., and Garcia, T. (2023). Análise de sentimentos em turmas de programação com vistas ao apoio à permanência estudantil. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 1329–1340, Porto Alegre, RS, Brasil. SBC.

Raabe, A. L. A. (2005). Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos. In *XIII Workshop de Educação em Computação*. SBC.

Ramos, I., Gadelha, B., Ramos, D., Oliveira, D., Mello, R., and Oliveira, E. (2023). Codegraph: Uma ferramenta de identificação e visualização de trilhas de aprendizagem no ensino de programação. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 451–462, Porto Alegre, RS, Brasil. SBC.

Sanches, W. M., Ferreira, F. Z., Evald, P. J., Vargas, A. P., Bez, J. L., and Botelho, S. S. d. C. (2023). Aprimorando a experiência de aprendizado em ambientes online massivos: o papel dos sistemas de recomendação. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 164–174. SBC.

Zhao, W. X., Zhang, W., He, Y., Xie, X., and Wen, J.-R. (2018). Automatically learning topics and difficulty levels of problems in online judge systems. *ACM Trans. Inf. Syst.*, 36(3).

Zorzo, A. F., Nunes, D., Matos, E., Steinmacher, I., Leite, J., Araujo, R. M., Correia, R., and Martins, S. (2017). Referencias de formação para os cursos de graduaç ão em computação. *Sociedade Brasileira de Computação*.