CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Avaliação do Pensamento Computacional em Graduandos de Cursos de Computação: uma Disciplina de Programação faz Diferença?

Álekiss Manço de Mélo[1], Sislan D. Nascimento Ferreira[1],
Ana Liz Souto Oliveira[1], Wilkerson L. Andrade[2]

[1]Departamento de Ciências Exatas (DCX) –
Universidade Federal da Paraíba (UFPB)

[2]Laboratório de Prática de Software (SPLab) –
Universidade Federal de Campina Grande (UFPB)

{alekiss.melo,sislan.davys,analiz}@dcx.ufpb.br,
wilkerson@computacao.ufcg.edu.br

Abstract. *This work aims to collect evidence on whether Computational Thinking (CT) skills differ between freshman undergraduates (with no prior knowledge of programming) and second/third-semester undergraduates (after enrolled one or two programming classes) in Computing courses. Data from 105 undergraduates were collected using an instrument containing CT problem questions. Our findings suggest no significant difference in CT skills between groups. More studies need to be conducted to understand how programming affects CT development in undergraduate Computing courses and how to measure CT and programming reliably.*

Resumo. *O objetivo deste trabalho é coletar evidências se as habilidades do Pensamento Computacional (PC) diferem entre graduandos ingressantes (sem conhecimento prévio em programação) e graduandos do segundo/terceiro período (após cursarem uma ou duas disciplinas de programação) em cursos de Computação. Os dados de 105 graduandos foram coletados através de um instrumento diagnóstico com questões-problemas de PC. Nossos resultados sugerem que não há diferença significativa nas habilidades de PC entre os grupos. Concluímos que mais estudos precisam ser conduzidos para entender como a programação afeta o desenvolvimento de PC na graduação de cursos de Computação e como mensurar PC e programação de forma confiável.*

1. Introdução

A avaliação da aprendizagem é um recurso necessário tanto na prática docente como nas pesquisas científicas. Melo (2020) faz um recorte de três modalidades principais de avaliação: a diagnóstica, a formativa e a somativa. A avaliação diagnóstica ocorre em um momento pontual, por exemplo, no início de um curso ou de um novo ciclo de conteúdo, com o intuito de detectar o estado de conhecimento prévio e erros dos estudantes. A formativa é empregada no decorrer da ministração do conteúdo, indicando que os estudantes estão se modificando em direção aos objetivos de aprendizagem. Por último, a avaliação somativa tem sua função no final de um curso ou ciclo de conteúdos

na intenção de classificar, e pode utilizar uma ou mais estratégias avaliativas ao longo do tempo.

A medição de habilidades cognitivas, independente da modalidade de avaliação, trás desafios sobre como inferir essas medidas. Diferente de medições como tempo e velocidades, as quais podem ser quantizadas numericamente por um instrumento, as habilidades cognitivas podem ser conceituadas por meio de construto e só depois construído um instrumento para medi-las. Um construto pode ser entendido como uma habilidade/capacidade que não podemos medir diretamente e por isso precisamos de um instrumento para estimar sua medida [Hutz *et al*, 2015].

No ensino e na pesquisa em Computação, um construto que tem grande destaque na última década é o Pensamento Computacional. Na visão da percursora do termo, pensamento computacional engloba reconhecer aspectos e habilidades da Computação no mundo de forma a aplicar ferramentas e técnicas para entender e raciocinar sobre sistemas e processos naturais, sociais e artificiais [Wing, 2006]. No contexto da pesquisa, podemos definir pensamento computacional como um construto que se expressam por meio de habilidades de resolver problemas explorando decomposição, abstração, pensamento algorítmico, generalização e avaliação [Csizmadia *et al*, 2015].

Na vertente de avaliação diagnóstica de pensamento computacional, podemos citar alguns instrumentos. O *CTtest* avalia habilidades de resolução de problemas relacionadas aos conceitos de Computação utilizando a sintaxe de lógica de programação para crianças de 12 a 14 anos e foi originalmente criado no idioma espanhol [Román-González *et al*, 2017]. Em uma vertente similar, mas em português, o *CT puzzle test* é uma ferramenta em desenvolvimento para avaliar pensamento computacional direcionada aos estudantes do 9º ano do ensino fundamental [Raabe *et al,* 2020]. Pensado para o público da educação superior, Oliveira e Pereira (2023) estão desenvolvendo um artefato que engloba um conjunto de habilidades e definições, evidências, e uma escala a fim de verificar o exercício o pensamento computacional em estudantes de graduação.

Apesar das abordagens existentes, há poucos instrumentos em português para avaliar habilidades do pensamento computacional, principalmente voltados a estudantes de graduação [Araujo *et al*, 2016; Avila *et al*, 2017]. Mais especificamente, embora haja evidências que programação auxilia no desenvolvimento de pensamento computacional, ainda não sabemos seus efeitos em cursos de graduação em Computação, nem como mensurá-lo de maneira apropriada [da Silva, & Falcão, 2020]. Assim, este estudo faz parte de uma pesquisa mais ampla que deseja investigar os efeitos do ensino de programação em cursos superiores de Computação e suas relações com o desenvolvimento de habilidades do pensamento computacional nos estudantes.

Nesse contexto, o objetivo deste trabalho exploratório é coletar evidências que as habilidades do pensamento computacional diferem entre graduandos ingressantes (sem conhecimento prévio em programação) e graduandos do segundo/terceiro período (após cursarem uma disciplina de introdução à programação com Python) em cursos de Computação. Assim, desejamos responder a seguinte questão de pesquisa: (QP1) Há diferença significativa quando se compara o desempenho em resolver problemas envolvendo pensamento computacional entre estudantes ingressantes e estudantes do segundo/terceiro período de cursos de Computação? (QP2) Qual o tamanho do efeito quando se compara o desempenho em resolver problemas envolvendo pensamento

computacional entre estudantes ingressantes e estudantes do segundo/terceiro período de cursos de Computação?

Para atingir esse objetivo e responder as questões de pesquisa, realizamos um estudo envolvendo 105 estudantes de graduação de dois cursos de Computação. Utilizamos, como instrumento para avaliação diagnóstica, questões-problemas que exploram habilidades do pensamento computacional utilizadas em um estudo prévio realizado por Mooney e Lockwood (2020). Para análise dos dados, utilizamos estatística descritiva e inferencial, bem como discutimos os resultados considerando o estado da arte em avaliação e mensuração de habilidades cognitivas.

O restante do trabalho está organizado da seguinte forma: Na Seção 2 são apresentadas as habilidades do pensamento computacional adotadas neste trabalho. Na Seção 3 são discutidos os trabalhos relacionados. Na Seção 4 são descritos os materiais e métodos. Na Seção 5 são apresentados e discutidos os resultados. Por último, na Seção 6 são apresentadas as considerações finais e trabalhos futuros.

2. Pensamento Computacional e suas habilidades como construtor

O Pensamento Computacional é um processo cognitivo envolvendo raciocínio lógico e habilidades comuns na Computação no qual os problemas são resolvidos. Csizmadia et al (2015) advogam por cinco habilidades essenciais: pensamento algorítmico, decomposição, generalização, abstração e avaliação.

Pensamento algorítmico é uma forma de chegar a uma solução por meio de uma definição clara das etapas ordenadas. O pensamento algorítmico é a capacidade de pensar em termos de sequências e regras como forma de resolver problemas ou compreender situações [Brackmann *et al*, 2019]. É uma habilidade fundamental que os alunos desenvolvem quando aprendem a criar programas. **Decomposição** é uma maneira de pensar sobre os artefatos em termos de suas partes componentes. As peças podem em seguida ser compreendidas, resolvidas e avaliadas separadamente. Isso torna os problemas complexos mais fáceis de resolver, por meio de situações que pode ser melhor. Por meio da decomposição da tarefa original, cada parte menor pode ser resolvida e integrada posteriormente no processo.

Generalização está associada à identificação de padrões, semelhanças e conexões, e à exploração desses recursos. É uma maneira de resolver novos problemas rapidamente com base em soluções de problemas anteriores. Isso permite ao indivíduo construir nova experiência. Fazer perguntas do tipo: “Isso é semelhante a um problema que já resolvi?” e “como é diferente de problemas que já resolvi?” são importantes [Csizmadia *et al*, 2015]. Assim pode-se reconhecer padrões de solução de problemas da mesma classe e generalizar formas de resolver. Por exemplo, algoritmos que resolvem problemas específicos podem ser adaptados para resolver classe de problemas semelhantes. Então, sempre que um problema dessa classe é encontrado, a solução (o algoritmo) pode ser aplicada.

Abstração torna os problemas mais fáceis de pensar. Abstração é o processo de fazer um artefato mais compreensível através da redução de detalhes desnecessários. Um exemplo clássico é a representação através de mapas turísticos: o mapa mostra os pontos que são interessantes para visitação turística e não coloca informações detalhadas de outras coisas. É uma representação que contém precisamente as informações necessárias para planejar visitas turísticas. A habilidade em abstração está

em escolher o detalhe certo para esconder, de forma que o problema se torne mais fácil, sem perder tudo o que é importante. **Avaliação** é o modo de garantir o recurso mais adequado para uma solução. Para isso, propriedades de soluções precisam ser avaliadas. Perguntas que podem facilitar o processo são: os passos para atingir a solução estão corretos? São eficientes o suficiente? Usam recursos economicamente? São fáceis de usar? Promovem uma experiência apropriada? Compensações precisam ser feitas, já que raramente há uma única solução ideal para todas as situações.

3. Trabalhos Relacionados

O trabalho de Mooney e Lockwood (2020) descreve o acompanhamento e avaliação de alunos do primeiro ano de graduação de Computação no tocante a habilidades de programação, pensamento computacional e concepções gerais do curso de Computação. Eles aplicaram um pré-teste e pós-teste baseado em questões do Desafio Bebras para avaliarem habilidades do PC nos estudantes no início e no final desse primeiro semestre letivo. Os autores chegaram à conclusão de que os alunos que tiveram bons resultados em matemática e conhecimento prévio em programação tiveram um desempenho melhor em comparação com o grupo que não tiveram bom desempenho em matemática e conhecimento prévio em programação.

Computational Thinking Test (CTtest) é um teste de CT de múltipla escolha on-line, baseado em conceitos de Computação relacionados com lógica de programação (Román-González et al, 2017). Embora não exija conhecimento prévio em programação, as questões abordam soluções de problemas envolvendo aplicação de sequências lógicas, loops, iterações, condicionais e funções. O público-alvo principal são adolescentes entre 12 e 14 anos. O teste está disponível nos idiomas espanhol e inglês. Uma tradução e adaptação para o Português do CTtest foi realizado por Brackmann et al (2019). Eles aplicaram o teste com o objetivo de averiguar a eficácia de intervenções de oficinas de PC por meio de atividades desplugadas.

Diferentemente de um teste de múltipla escolha, o CT Puzzle Test se propõe a realizar uma avaliação baseada em puzzle aplicado de forma *on-line* (Raabe *et al*, 2020). Ele foi construído inspirado em exercícios clássicos de resolução de problemas, com níveis de dificuldade crescente. O teste não exige conhecimento prévio de programação nem de outro conteúdo. Além de mensura quatro habilidades de PC (pensamento algorítmico, abstração, reconhecimento de padrões e decomposição), o teste também registra interações como tempo despendido, quantidade de clique ou comandos ou dicas utilizadas.

4. Materiais e Métodos

Nesta Seção apresentamos os participantes da pesquisa, o instrumento utilizado e a forma de coleta e análise dos dados.

4.1. Participantes

Os participantes da pesquisa somam no total de 105 estudantes da Universidade Federal da Paraíba, regularmente matriculados nos cursos de Licenciatura em Ciência da Computação e Bacharelado em Sistemas de Informação em 2021. Dessa amostra, 63 estudantes eram ingressantes (primeiro período) matriculados na disciplina de Introdução à Programação; 21 estudantes eram do segundo período (P2) matriculados

na disciplina de Linguagem de Programação; e por fim, 21 estudantes do terceiro período (P3) matriculados na disciplina de Programação Orientada à Objetos.

A disciplina de Introdução à Programação é ministrada empregando a linguagem de programação Python em todas as turmas e é pré-requisito para a disciplina de Linguagem de Programação. Portanto, os estudantes que estão no segundo e terceiro período, obrigatoriamente, possuem conhecimento introdutório em Python. Já as disciplinas Linguagem de Programação e Programação Orientada à Objetos são ministradas na linguagem Java. A disciplina de Linguagem de Programação (do segundo período P2) é pré-requisito para Programação Orientada à Objetos (do terceiro período P3).

4.2. Instrumento

O instrumento adotado nesta pesquisa para avaliação diagnóstica do pensamento computacional foi um conjunto de questões-problemas que exploram habilidades do pensamento computacional para sua resolução. Essas questões foram traduzidas e adaptadas da pesquisa de Mooney e Lockwood (2020) realizadas com estudantes de graduação ingressantes em cursos de Computação na Irlanda. Esse instrumento contém questões que fazem parte de um concurso de pensamento computacional chamado Bebras. Bebras é uma comunidade internacional de professores e pesquisadores fundada na Lituânia em 2004, que tem por objetivo promover pensamento computacional nas escolas. As questões produzidas pela comunidade têm como enredo uma família de castores que simulam problemas do cotidiano, dando uma visão mais lúdica. As questões envolvem um problema que aborda um conceito de Computação que pode ser solucionado utilizando habilidades do pensamento computacional. Entretanto, não é necessário conhecimento prévio em Computação ou programação para resolver o problema.

As questões utilizadas estão disponíveis na íntegra em (Link para Google Drive). Portanto, o instrumento contém 11 (onze) questões de múltipla escolha com uma única resposta correta.

Após a escolha do instrumento, pelo fato das questões estarem originalmente na língua inglesa, executamos o processo de tradução e adaptação para o português. Pesquisadores externos à pesquisa auxiliaram no processo verificação/ajustes da tradução. As questões foram adaptadas para serem exibidas no *Google form* a fim de facilitar a coleta e armazenamento das respostas.

Ressaltamos que após o processo de tradução, realizamos um teste piloto com uma amostra reduzida. Verificamos se as questões estavam inteligíveis, se o instrumento apresentava alguma falha e se os dados (respostas) estavam sendo registradas adequadamente. As correções foram realizadas e as questões ficaram prontas para coleta de dados.

4.3. Coleta e análise de Dados

A coleta de dados foi realizada no segundo semestre de 2021 de forma remota, devido as condições de distanciamento social impostos pela pandemia da COVID-19. A universidade estava funcionando na modalidade de Ensino Remoto Emergencial, devido as medidas sanitárias impostas pela Organização Mundial de Saúde. Considerando esse contexto particular, o pesquisador principal, com consentimento dos professores, visitou

as turmas virtuais nas quais os estudantes de Introdução à Programação (ingressantes), Linguagem de Programação (segundo período - P2) e Programação Orientada à Objetos (terceiro período – P3) estavam matriculados. Ele explicou aos estudantes o objetivo e contribuições da pesquisa, bem como as questões éticas envolvendo os dados coletados. Depois, convidou os estudantes a continuarem na sala virtual para responderem o instrumento de forma voluntária. Posteriormente, foi disponibilizado o link do instrumento e estabelecido o tempo de até 45 minutos para a sua resolução.

Após a coleta dos dados em todas as turmas virtuais, as questões foram corrigidas e os resultados tabulados em planilha. Entendemos que no nosso contexto de pesquisa, a variável independente é ter cursado disciplina de programação introdutória e a variável dependente é o número de acertos no instrumento. Para análise dos dados, utilizamos estatísticas descritiva e inferencial para responder às questões de pesquisa e discutimos os resultados e limitações do estudo na próxima seção.

5. Resultados e Discussão

A coleta de dados contém respostas de uma amostra composta por 105 alunos, sendo 63 das disciplinas de Introdução à Programação, 21 das turmas de Linguagem de Programação (P2) e 21 das turmas de Programação Orientada à Objetos (P3). Quanto a caracterização da amostra, 82.9% declararam ser do sexo masculino, 14.3% do sexo feminino e 2,9% preferiram não declarar. A média de idade foi de 20,5 anos e 69,5% dos estudantes residiam em pequenos centros urbanos (que não é a capital do estado).

Na Figura 1 observamos o gráfico de barras indicando a quantidade de acertos dos estudantes ingressantes ($n = 63$) e estudantes do segundo/terceiro período ($n = 42$). Como a quantidade de estudantes difere entre os dois grupos, optamos por mostrar a porcentagem de acertos para facilitar a comparação visual. Assim, notamos que os estudantes do P2/P3 apresentam uma ligeira maior quantidade de acertos em quase todas as questões, com exceção da questão 5 (cinco) e 8 (oito), em relação aos alunos ingressantes. Em particular, na questão 10 (dez), a quantidade de acerto dos estudantes do P2/P3 foi mais de 20% superior ao dos ingressantes.

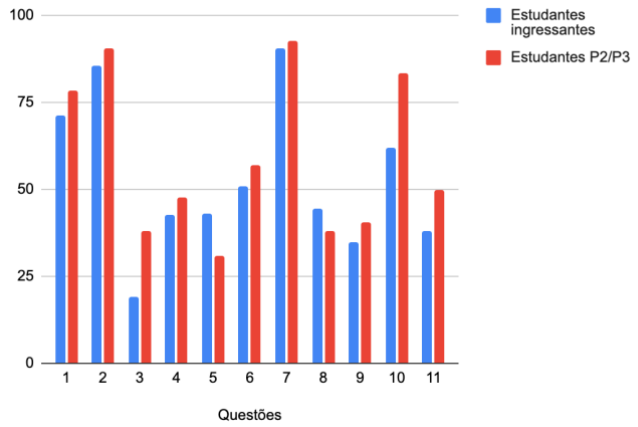

Figura 1. Número de acertos das questões (em porcentagem) dos estudantes ingressantes comparado com estudantes do P2/P3

5.1 (QP1) Há diferença significativa quando se compara o desempenho em resolver problemas envolvendo pensamento computacional entre estudantes ingressantes e estudantes do segundo/terceiro período de cursos de Computação?

Para responder a primeira questão de pesquisa, inicialmente, utilizamos o teste de Shapiro-Wilk, com o objetivo de avaliar se a distribuição das amostras seguia uma distribuição normal ou não. Consideramos como hipótese nula "***H0**: Os acertos dos estudantes no instrumento seguem a distribuição normal"* e como hipótese alternativa "***Ha**: Os acertos dos estudantes no instrumento não seguem a distribuição normal"*. Obtivemos com o teste Shapiro-Wilk, com nível de significância de 5%, os seguintes valores para as amostras: acertos dos Ingressantes: *p-valor = 0.528*; acertos dos estudantes do segundo/terceiro: *p-valor = 0.821*. Como todos os *p-valores* foram superiores a *0.05*, consideramos a hipótese nula como aceita, e interpretamos que as amostras seguem uma distribuição normal, com nível de significância de 5%.

Para comparar as médias de acertos amostrais, foi usado o teste *T de Student*. A hipótese nula é "***H0**: A média de acertos dos estudantes ingressantes é igual a média de acertos dos estudantes do segundo/terceiro período"* (μIngressantes = μP2P3) e a hipótese alternativa é "***Ha***: *A média de acertos dos estudantes ingressantes é diferente da média de acertos dos estudantes do segundo/terceiro período* (μIngressantes ≠ μP2P3). Para o teste *T de Student*, consideramos que as duas amostras são independentes e seguem distribuição normal, adotamos o nível de significância de 5%. Como está sendo verificado se as médias de acertos diferem, o teste executado é bicaudal.

Assim, o resultado do teste *T de Student* apontou um *p-valor = 0,13*. Como o *p-valor* foi maior que 0,05, não podemos rejeitar a hipótese nula, ou seja, não é possível afirmar que as médias entre esses dois grupos diferem. Sendo assim, segundo o teste T, podemos considerar a hipótese alternativa, que as diferenças de acertos dos estudantes dos dois grupos não são estatisticamente significativas, considerando nível de significância de 5%.

Posteriormente, por curiosidade, investigamos se existia diferença nas médias de acerto entre os estudantes do segundo e terceiro período. Lembrando que os estudantes do *P2* são os alunos do segundo período cursando a disciplina de Linguagem de Programação e os estudantes do *P3* são os alunos do terceiro período cursando a disciplina de Programação Orientada à Objetos. O teste *T de Student* também foi utilizado para comparar as médias desses dois grupos. A hipótese nula é "***H0**: a média de acerto dos estudantes do P2 é igual a média de acertos dos estudantes do P3"* (μP2 = μP3). Como hipótese alternativa, temos que "***Ha**: a média de acerto dos alunos do P2 é diferente a média de acertos dos alunos do P3"* (μP2 ≠ μP3). O resultado do teste bicaudal apontou *p-valor = 0,57*, com nível de significância de 5%. Dessa forma, podemos aceitar a hipótese nula e considerar que temos indícios para afirmar que não existe diferença significativa entre os dois grupos, considerando nível de significância de 5%.

A Figura 2 apresenta o gráfico de barras de acertos dos estudantes do P2 (do segundo período, *n = 21*) e estudantes do P3 (do terceiro período, *n = 21*). Nesse

gráfico, optamos por usar o número real de participantes, uma vez que ambos os grupos possuem 21 estudantes. Realizando uma análise de percepção visual, notamos que existe equiparidade entre a proporção de acertos dos dois grupos. Portanto, a análise visual do gráfico corrobora com os resultados obtidos com o teste de hipótese, de que não há diferença significativa na média de acertos entre os estudantes do P2 e P3.

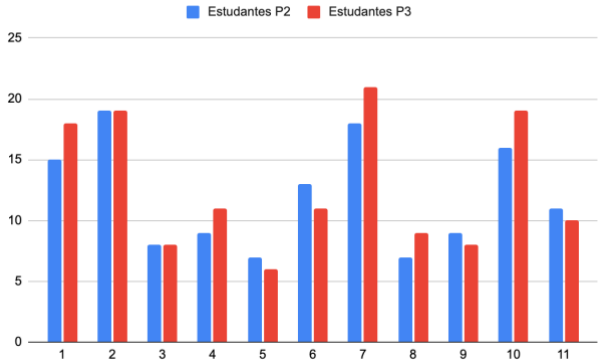

Figura 2. Quantidade de acertos das questões dos estudantes do segundo período (P2) e terceiro período (P3)

5.2 (QP2) Qual o tamanho do efeito quando se compara o desempenho em resolver problemas envolvendo pensamento computacional entre estudantes ingressantes e estudantes do segundo/terceiro período de cursos de Computação?

Para medir o tamanho do efeito da diferença entre as médias de acertos entre os estudantes ingressantes e os estudantes do segundo/terceiro período, utilizamos o D de Cohen. O resultado obtido com o tamanho do efeito foi de $d = 0,2999$. Para esse valor, podemos considerar a magnitude de efeito como pequena. Esse resultado corrobora com a decorrência de não haver diferença significativa das médias de acertos dos estudantes dos dois grupos, encontrado investigação da primeira questão de pesquisa.

5.3 Discussão e limitações do estudo

No início da nossa pesquisa, acreditávamos que encontraríamos indícios para afirmar que haveria diferenças nas médias de acertos entre estudantes ingressantes e estudantes do segundo/terceiro período. Nossas expectativas estavam atreladas a estudos que apontam que ensino de programação desenvolve habilidades de pensamento computacional. O grupo P1 (ingressantes) responderam o teste antes do início do ensino de introdução à programação com a linguagem Python e os grupos P2 e P3 já possuíam conhecimento em Python e estavam aprendendo Java. Assim, acreditávamos que o fato dos estudantes do P2/P3 conhecerem duas linguagens de programação pudessem interferir no número de acerto das questões. Entretanto, nossos resultados estão alinhados aos resultados do estudo de Mooney e Lockwood (2020). Os autores apontaram que não há diferença significativa entre as médias de acerto no pré-teste e pós-teste aplicado antes e depois dos alunos ingressantes de Computação cursarem a disciplina de introdução à programação na Irlanda.

Por um lado, podemos pensar que habilidades do pensamento computacional não são exclusivas da Computação. As pessoas já desenvolvem de alguma maneira, por exemplo, habilidades seguir passos (algoritmos) a partir de outras experiências

(escolares ou não). A execução de operações matemáticas é um exemplo de algoritmo que crianças e adolescentes aprendem na escola. Por outro lado, é de se esperar que o estudo formal de programação aprimore a habilidade de seguir e (criar) algoritmos.

Nossos resultados apontaram que ainda é incipiente entender como o pensamento computacional está relacionado ao aprendizado formal de programação no ensino superior. Os autores da Silva & Falcão (2020) reforçam nossos achados afirmando que ainda é necessário investigar mais a fundo como desenvolver e avaliar habilidades de pensamento computacional em estudantes de cursos superiores a fim de não tratar pensamento computacional e ensino de programação como sinônimos. Todavia, creditamos que competências de pensamento computacional são desenvolvidas a partir da aprendizagem de programação, mas precisamos investigar como isso acontece. O estudo de Oliveira e Pereira (2023) podem trazer mais contribuições quando aplicado alinhado ao ensino de programação.

Por último, ressaltamos que nossa pesquisa possui limitações e ameaças à validade. Embora o número da amostra possa ser considerado pequeno, tanto o teste *T de Student* como D de Cohen podem ser empregados para amostra acima de 50 participantes sem perda de significância. O instrumento empregado para avaliação diagnóstica do pensamento computacional não possui validação. Mesmo assim, as questões produzidas pelo Bebras têm sido investigadas amplamente como uma forma de aplicar habilidades do pensamento computacional na resolução dos problemas (Román-González *et al*, 2019).

6. Considerações Finais

Neste trabalho investigamos se habilidades do pensamento computacional diferem entre graduandos ingressantes e graduandos do segundo/terceiro período (após cursarem uma disciplina de introdução à programação com Python) em cursos de Computação. Nós traduzimos e aplicamos um instrumento para realizar uma mensuração de pensamento computacional por meio de acertos de questões-problemas.

Nossos resultados apontaram que não podemos afirmar que há diferença significativa quando se compara o desempenho em resolver problemas envolvendo pensamento computacional entre estudantes ingressantes e estudantes do segundo/terceiro período de cursos de Computação, mesmo que esse último grupo tenha conhecimento de duas linguagens de programação. Complementando esse achado, encontramos que o tamanho do efeito é pequeno quando comparamos as médias de acerto entre os grupos. Apesar disso, ressaltamos que nossos resultados não podem ser generalizados devido ao tamanho e representatividade da amostra, bem como a não validação do instrumento utilizado. Mais investigação precisa ser conduzida para compreender melhor como se desenvolve as habilidades de pensamento computacional em graduandos nos cursos de Computação.

Nessa vertente, nossa pesquisa contribui para apontar a necessidade de instrumentos para avaliação diagnóstica de habilidades de pensamento computacional no ensino superior. Também acreditamos que competências de PC são desenvolvidas a partir da aprendizagem de programação, mas que dentro de cursos superiores de Computação, ainda não temos a clareza como o desenvolvimento dessas habilidades estão relacionadas a aprendizagem de programação em diferentes níveis, bem como outras disciplinas.

Como trabalhos futuros, planejamos realizar nova coleta de dados com estudantes de Computação de outras universidades. Um outro estudo poderia ser conduzido com alunos de cursos na área de matemática e engenharia, e conduz estudos comparativos para investigar a relação de pensamento computacional e o ensino da matemática e engenharia em cursos superiores. Além disso, não abordamos as habilidades do pensamento computacional separadamente, apenas como um construto só. Então, para estudos futuros, devemos investigar como seria possível mensurar habilidades separadamente.

Referências

Avila, C., Cavalheiro, S., Bordini, A., Marques, M., Cardoso, M., & Feijo, G. (2017). Metodologias de Avaliação do Pensamento Computacional: uma revisão sistemática. In Simpósio Brasileiro de Informática na Educação - SBIE, 28(1), 113.

Araújo, A. L., Andrade, W., & Guerrero, D. (2016). Um mapeamento sistemático sobre a avaliação do pensamento computacional no Brasil. In Anais dos Workshops do Congresso Brasileiro de Informática na Educação, volume 5, page 1147.

Brackmann, C. P., Caetano, S. V. N., & da Silva, A. R. (2019). Pensamento Computacional Desplugado: ensino e avaliação na educação primária brasileira. RENOTE, 17(3), 636-647.

Csizmadia, A. et al (2015). Computational thinking a guide for teachers. Computing at school, 2015. Disponível em: Computing at School. Acesso em: 26 set. 2021.

da Silva, E., & Falcão, T. (2020). O Pensamento Computacional no Ensino Superior e seu Impacto na Aprendizagem de Programação. In Anais do XXVIII Workshop sobre Educação em Computação, (pp. 171-175). Porto Alegre: SBC. doi:10.5753/wei.2020.11152

Guarda, G. F., & Pinto, S. C. C. (2020). Dimensões do Pensamento Computacional: conceitos, práticas e novas perspectivas. In Anais do XXXI Simpósio Brasileiro de Informática na Educação (pp. 1463-1472).

Hutz, C. S., Bandeira, D. R., and Trentini, C. M. (2015). Psicometria. Artmed Editora.

Howarth, L. et al (2014). UK Bebras Computational Thinking Challenge. Disponível em: Welcome - UK Bebras. Acesso em: 26 set. 2021.

Melo, Ronaldo Silva (2020). *Conceitos e fundamentos da avaliação*. Natal: SEDIS/UFRN.

Mooney, A. & Lockwood, J. (2020). The Analysis of a Novel Computational Thinking Test in a First Year Undergraduate Computer Science Course. Aishe-j, 2020. Disponível em: https://ojs.aishe.org/index.php/aishe-j/article/view/420. Acesso em: 26 set. 2022.

Oliveira, C. M., & Pereira, R. (2023). Coleta de Evidências do Exercício do Pensamento Computacional no Ensino Superior em Computação: um artefato de apoio. In *Anais do III Simpósio Brasileiro de Educação em Computação* (pp. 300-309). SBC.

Raabe, A., Viana, C., & Calbusch, L. (2020). CT Puzzle Test: Em direção a uma avaliação interativa do pensamento computacional. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação* (pp. 1683-1692). SBC.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in human behavior, 72, 678-691.

Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining assessment tools for a comprehensive evaluation of computational thinking interventions. Computational thinking education, 79-98.

Wing, J. 2006. Computational thinking. Commun. ACM, 49. p.33-35.