Clusterização de soluções de exercícios de programação: um mapeamento sistemático da literatura

Rafaela Melo[1], **Marcela Pessoa**[2], **David Fernandes**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (ICOMP - UFAM)
Caixa Postal 15.064 – 91.501-970 – Manaus – AM – Brazil

[2]Escola Superior de Tecnologia – Universidade do Estado do Amazonas (EST/UEA)
Manaus - AM - Brasil

{rmelo,david}@icomp.ufam.edu.br, msppessoa@uea.edu.br

Abstract. *In programming courses, students may adopt similar strategies when solving exercises. Clustering students' codes according to the strategies adopted can provide valuable insights for teachers. However, conducting this clustering manually is laborious, and therefore some works in the literature have explored approaches to cluster codes automatically. In view of this, this paper presents a Systematic Literature Mapping (SLM) on the clustering of programming exercise solutions. Twenty articles were identified, where the motivations for using clustering included generating personalized feedback and identifying common errors among students.*

Resumo. *Em disciplinas de programação, os estudantes podem adotar estratégias semelhantes ao solucionar exercícios. Agrupar os códigos dos alunos de acordo com as estratégias adotadas pode fornecer insights valiosos para os professores. No entanto, conduzir esse agrupamento de forma manual é trabalhoso, e por isso alguns trabalhos da literatura exploraram abordagens para agrupar códigos de forma automática. Diante disso, este artigo apresenta um Mapeamento Sistemático da Literatura (MSL) sobre a clusterização de soluções de exercícios de programação. Foram identificados 20 artigos, onde as motivações para o uso de clusterização incluíram a geração de feedback personalizado e a identificação de erros comuns entre os estudantes.*

1. Introdução

Em cursos de computação, um único exercício de programação pode ser resolvido de diversas formas [Joyner et al. 2019]. Por exemplo, um exercício de busca binária em vetores pode ser solucionado através de estratégias iterativas ou recursivas. Compreender as diferentes soluções desenvolvidas para um determinado problema pode ser útil do ponto de vista pedagógico, possibilitando aos professores fornecer *feedbacks* adequados e personalizados aos estudantes [Glassman et al. 2015]. Entretanto, compreender os códigos e identificar as diferentes estratégias para resolver os exercícios não é uma tarefa fácil.

Atualmente tem crescido o número de instituições que usam sistemas juízes online para automatizar a correção dos códigos dos alunos. Tais sistemas são capazes de avaliar automaticamente a corretude dos códigos submetidos através da comparação dos resultados produzidos pelos programas com os resultados esperados, utilizando casos

DOI: 10.5753/sbie.2024.242403

de teste pré-definidos [Wasik et al. 2018]. Uma das consequências do uso de sistemas juízes on-line é a possibilidade de tornar o professor dependente do *feedback* fornecido pelo sistema, diminuindo o seu contato com os códigos desenvolvidos pelos alunos [Barbosa et al. 2023]. Entretanto, ao visualizar ou analisar um código, é possível verificar lacunas existentes na compreensão dos estudantes e, assim, realizar intervenções conforme a necessidade [Koivisto and Hellas 2022].

Diante disso, pesquisas têm buscado formas de explorar as diversas soluções que existem para resolver um único problema [Glassman et al. 2015, Effenberger and Pelánek 2021], e fornecer *feedback* personalizado para os alunos [Rahman et al. 2021, Head et al. 2017]. Uma das abordagens mais utilizadas nesse contexto é o uso de técnicas de *clustering* para agrupar soluções semelhantes, o que permite descobrir padrões existentes em cada grupo [Luo and Zeng 2016].

Com o intuito de identificar os métodos usados na literatura para agrupar códigos e as motivações por trás do uso de tais técnicas, este artigo apresenta um Mapeamento Sistemático da Literatura (MSL) sobre a clusterização de soluções de exercícios de programação. O restante do artigo está organizado como segue: a Seção 2 aborda conceitos sobre clusterização, a Seção 3 apresenta o protocolo do mapeamento, a Seção 4 discorre sobre os resultados e na Seção 5 estão as considerações finais do trabalho.

2. Métodos de *clustering*

Clustering é uma técnica de aprendizagem não-supervisionada, ou seja, que envolve a análise de dados sem rótulos ou categorias predefinidas. Trata-se de uma técnica que tem como foco agrupar dados similares em "*clusters*" [Xu and Tian 2015], onde as instâncias dentro de um mesmo *cluster* devem ser o mais similares possível e, as instâncias em diferentes *clusters* devem ser o mais dissimilares quanto possível [Jain and Dubes 1988].

Segundo Xu e Tian [2015], os algoritmos tradicionais de *clustering* dividem-se em nove categorias, dentre elas estão: i) algoritmos baseados em partição, que consideram o centro dos pontos de dados como o centro do *cluster* correspondente (e.g., k-means [MacQueen et al. 1967], k-medoids [Kaufman and Rousseeuw 2009]); ii) algoritmos baseados em hierarquia, onde é construído um relacionamento hierárquico entre os dados para realizar o agrupamento (e.g., *hierarquical clustering* [Ward Jr 1963]); iii) algoritmos baseados em densidade, onde dados que estão em uma região de alta densidade no espaço de dados pertencem ao mesmo *cluster* (e.g., DBSCAN [Ester et al. 1996], OPTICS [Ankerst et al. 1999]); e, iv) algoritmos baseados em modelos, onde é selecionado um modelo específico para cada *cluster* e encontra-se o melhor ajuste para esse modelo (e.g., GMM (*Gaussian Mixture Model*) [Bishop 2006]).

Existem, ainda, algoritmos de *clustering* considerados modernos [Xu and Tian 2015], como: i) *affinity propagation*, que considera todos os pontos de dados como os potenciais centros de um *cluster* e, a partir disso, procura encontrar os exemplos que resultam na maior preferência total entre todos os pontos de dados [Xu and Tian 2015]; e, ii) *spectral clustering*, que utiliza autovalores e autovetores de uma matriz de similaridade para reduzir a dimensionalidade dos dados e identificar *clusters* [Von Luxburg 2007].

Considerando o contexto de agrupar soluções de exercícios de programação, que é o foco de pesquisa deste MSL, os algoritmos de *clustering* parecem ser mais adequa-

dos tanto que foram aplicados em diversos trabalhos na literatura, como será possível visualizar nas próximas seções.

3. Protocolo do Mapeamento Sistemático da Literatura (MSL)

Para conduzir este mapeamento foi utilizado um protocolo baseado em Kitchenham et al. [2022], que define os passos a serem seguidos para identificar, avaliar e interpretar estudos primários que podem ser relevantes dentro de um tópico de pesquisa específico. Esta seção apresenta as etapas do protocolo, bem como as questões de pesquisa, as estratégias de busca, os critérios de inclusão e exclusão, a criação da *string* de busca e o processo de seleção.

3.1. Questões de Pesquisa

Este MSL tem como objetivo identificar e analisar estudos da literatura que exploram estratégias de clusterização para agrupar soluções de exercícios de programação, com base na estratégia utilizada pelos estudantes. Dentro desse contexto, o foco principal deste trabalho é consolidar os métodos de *clustering* adotados nesses estudos, as estratégias de visualização de *clusters* usadas (quando houver) e quais as motivações para o uso de clusterização de soluções. Para tanto, foram definidas as seguintes questões de pesquisa (QPs):

- QP1: Quais são os métodos utilizados para agrupar os códigos dos alunos de acordo com as estratégias adotadas ao solucionar os exercícios de programação?
- QP2: São utilizadas técnicas para ajudar os professores na visualização dos agrupamentos de soluções? Se sim, quais?
- QP3: Quais são as motivações citadas por esses trabalhos para a adoção de métodos de agrupamento de soluções de programação?

3.2. String de busca

Para responder às QPs, primeiro foram selecionados alguns artigos de controle na literatura [Glassman et al. 2015, Effenberger and Pelánek 2021, Joyner et al. 2019, Luo and Zeng 2016, Koivisto and Hellas 2022]. Esses artigos serviram de base para a construção da *string* de busca, ao auxiliar na seleção de termos do títulos, palavras-chave e resumos dessa publicações. Posteriormente, eles também ajudaram na validação da *string* de busca, que deveria ser capaz de retornar todos os artigos de controle em seus resultados. A seguir são listadas as palavras selecionadas para compor a *string*:

- **Soluções de programação:** “code” OR “solution” OR “programming”;
- **Agrupamento:** “cluster” OR “clustering” OR “group”;
- **Contexto:** “online judge” OR “introductory programming” OR “programming education” OR “programming course” OR “cs1” OR “learning programming” OR “learn programming”.

3.3. Base de busca e critérios de inclusão e exclusão

A busca automatizada com a *string* de busca foi realizada na base de dados Scopus[1]. Essa base foi eleita por ser uma das maiores bases de dados de resumos da literatura, além

[1]https://www.scopus.com

de fornecer ferramentas avançadas de busca e ser de fácil acesso e uso. Os critérios de inclusão (CI) e exclusão (CE) foram definidos da seguinte forma:

Critérios de Inclusão: i) Artigos que propõem métodos para agrupar soluções de acordo com a estratégia de programação; ii) Artigos que abordam técnicas de *clustering* no contexto educacional.

Critérios de Exclusão: i) Não atender a todos os critérios de inclusão; ii) O artigo não ser um artigo completo (considerando para isso no mínimo 5 páginas); iii) A versão completa do artigo não estar disponível; iv) O idioma do artigo não ser inglês; v) O artigo estar duplicado; vi) Ser livro ou capítulo de livro, tese ou dissertação; vii) O estudo ser um estudo secundário (revisões, mapeamentos sistemáticos da literatura ou mesmo estudos apresentando revisões informais da literatura); viii) O estudo ser uma versão mais antiga de outro estudo já considerado; ix) O artigo ser sumário de anais de eventos científicos, descrição de um curso, editorial, resumo de palestra, *workshop* ou tutorial.

3.4. Processo de seleção

A condução desta etapa se baseou no processo de seleção realizado no MSL de Pessoa et al. [2023], onde foram realizados três filtros. No caso do presente trabalho, o primeiro filtro tratou da leitura do título, resumo e palavras-chave dos artigos resultantes da busca na base com a *string*. No segundo filtro, foi realizada uma leitura rápida dos artigos, e, por último, foi feita uma leitura minuciosa das publicações, onde os dados foram extraídos e documentados.

4. Resultados e Discussão

A definição do protocolo de revisão e da estratégia de busca foi conduzida por dois pesquisadores. A execução da *string* de busca na biblioteca digital selecionada resultou em 1528 artigos. Após a leitura do título e resumo de cada artigo (primeiro filtro), removendo as publicações que não atendiam a todos os critérios de inclusão ou que atendiam a pelo menos um critério de exclusão, restaram 51 artigos. Em seguida, as publicações passaram por uma leitura rápida, o que excluiu 15 artigos, e, por fim, foi realizada a leitura completa, onde foram retiradas aquelas que não atendiam aos critérios, restando 20 artigos.

Para reduzir o viés de somente um pesquisador, durante o processo de seleção, foi definida uma amostra aleatória de vinte publicações para que dois pesquisadores verificassem os critérios de inclusão e exclusão, com base no título e no resumo. Os resultados da seleção dos dois pesquisadores foram avaliados a partir do teste estatístico Kappa [Cohen 1960], para avaliar a concordância. O teste indicou que os pesquisadores convergiram estatisticamente (Kappa = 0,81) na seleção das publicações, aumentando a confiança e diminuindo o viés do pesquisador na inclusão dos artigos.

4.1. Contexto e linguagens de programação

Um dos aspectos analisados nas publicações foi a escolha das linguagens de programação utilizadas nos códigos dos alunos, e o contexto em que essas linguagens foram aplicadas. Como pode ser visto na Figura 1, as linguagens de programação com as quais os autores realizaram seus experimentos de clusterização foram: Python (6), Java (4), C (3), C# (2),

C++ (1), Ruby (1) e linguagem em blocos/visual (1). Três das publicações não especificaram qual a linguagem de programação usada pelos alunos. O artigo de Fu et al. [2021] foi o único a citar o uso de mais de uma linguagem (Java e C#), as demais publicações citaram apenas uma.

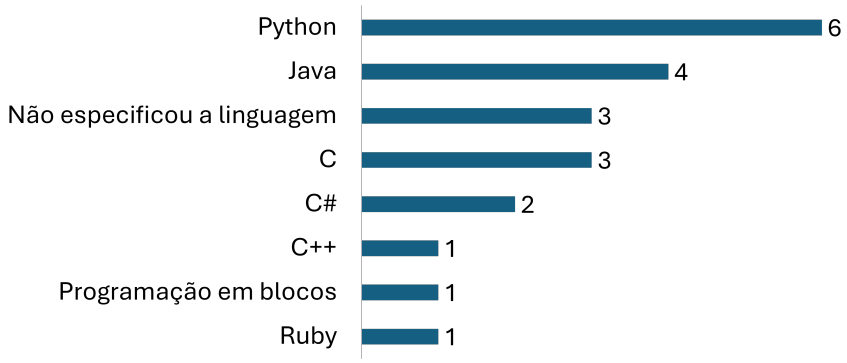


Figura 1. Linguagens de programação dos códigos.

Com relação ao contexto em que os algoritmos de clusterização foram aplicados, em onze publicações não foi especificado de qual disciplina se tratavam os códigos da base de dados. A disciplina Introdução à Programação foi abordada em oito trabalhos e, em um dos artigos, o foco foi a disciplina de Algoritmos e Estrutura de Dados. Referente às bases de dados, a distribuição foi da seguinte forma: Dataset avulso (6), Sistema de avaliação automatizada/Juiz on-line (4), Cursos on-line (3), Dataset conhecido (2), Sistema de gestão de aprendizagem (1), Ambiente de aprendizagem on-line (1), Ambiente de competição on-line (1) e Ambiente de programação em blocos (1).

No trabalho de Koivisto e Hellas [2022] foi apresentado o CodeClusters, um sistema de revisão manual de envios de tarefas de programação. Por ser uma proposta inicial, os autores não realizaram experimentos com nenhuma base de dados, mas informaram que a linguagem de programação aceita é o Java e que o sistema foi projetado para ser usado em conjunto com um sistema de avaliação automatizado/juiz on-line. Em Rahman et al. [2021], Kawabayashi et al. [2021], Gao et al. [2019] e Rahman et al. [2022], os autores usaram códigos capturados de juízes on-line para seus experimentos de clusterização.

Ainda sobre as fontes de dados, Beh et al. [2016], Rosales-Castro et al. [2016], Silva et al. [2023], Silva e Silla [2020], Barbosa et al. [2018] e Head et al. [2017] selecionaram códigos de exercícios de programação específicos para fazerem seus experimentos de clusterização, por isso foram classificados como “Dataset avulso”. Em Joyner et al. [2019], Yin et al. [2015] e Glassman et al. [2015], os dados foram capturados de cursos on-line como MOOC (*Massive Open Online Courses*) e edX. Já em Fu et al. [2021] e Comfébis e Schils [2016], os autores usaram dados provenientes do dataset do CodeHunt[2].

Os demais artigos usaram dados capturados de: sistema de gestão de aprendizagem [Lokkila et al. 2022], ambiente de aprendizagem on-line

[2]https://codehunt.cc/

[Effenberger and Pelánek 2021], ambiente de competição on-line (CodeForces[3]) [Luo and Zeng 2016] e ambiente de programação em blocos [Emerson et al. 2020].

4.2. Métodos de clusterização (QP1)

Entre as 20 publicações selecionadas, os autores optaram por diferentes estratégias para realizar o agrupamento de soluções. Como pode ser visto no gráfico da Figura 2, as publicações se distribuíram da seguinte forma: i) aplicou somente um método de clusterização conhecido da literatura (dez artigos); ii) propôs e utilizou um método próprio (cinco artigos); iii) usou dois métodos da literatura (dois artigos); iv) utilizou cinco métodos distintos da literatura (um artigo); v) usou quatro métodos da literatura (um artigo); e, vi) usou uma ferramenta de *cluster* (um artigo).

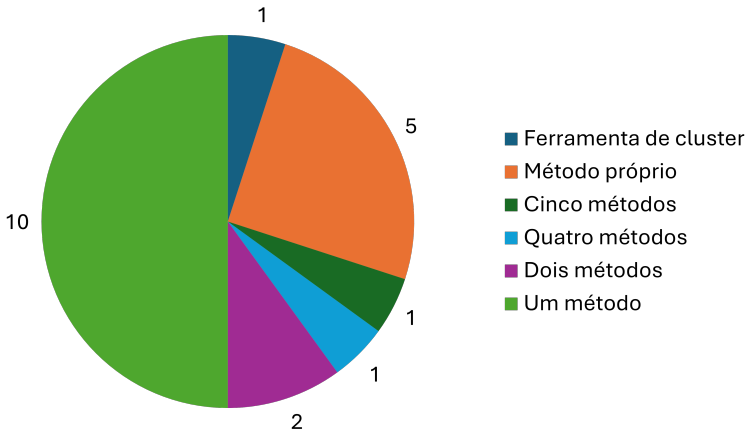


Figura 2. Distribuição de métodos por publicação.

Com relação aos algoritmos de clusterização, foram identificados dez que são conhecidos da literatura (K-means, MK-means, K-medoids, *Hierarchical*, *Spectral*, OPTICS, DBSCAN, HDBSCAN, *Affinity Propagation*, *Bayesian Gaussian Mixture Models*). A Tabela 1 mostra uma visão geral da distribuição das publicações entre os métodos identificados, bem como a categoria a qual o método faz parte, sendo: i) baseado em partição; ii) baseado em hierarquia; iii) baseado na teoria espectral; iv) baseado em densidade; v) baseado em afinidade; e, vi) baseado em modelo.

Koivisto e Hellas [2022], Yin et al. [2015], Rosales-Castro et al. [2016] e Combéfis e Schils [2016] foram publicações que aplicaram mais de um método de clusterização em suas pesquisas. O algoritmo K-means foi utilizado em 6 artigos, sendo que em um dos trabalhos os autores aplicaram técnicas para encontrar o melhor valor de K [Rahman et al. 2021], ou seja, o melhor número de *clusters*. Essa abordagem pode ser importante, visto que não há como saber quantos tipos de estratégias diferentes existem dentro de determinado conjunto de dados. Aplicar algoritmos de *cluster* como OPTICS, DBSCAN e HDBSCAN, que não exigem que o valor de K seja informado, pode ser mais indicado em problemas como esse, onde não se conhece todas as classes existentes no conjunto [Yin et al. 2015, Koivisto and Hellas 2022].

Quanto aos trabalhos que apresentaram métodos próprios, em Gao et al. [2019], os autores se basearam no algoritmo *Spectral Clustering* e propuseram um novo algoritmo

[3]https://codeforces.com/

Tabela 1. Métodos de clusterização utilizados nas publicações selecionadas.

Categoria	Método	Quantidade	Publicações
Baseado em partição	K-means	6	[Lokkila et al. 2022], [Koivisto and Hellas 2022], [Kawabayashi et al. 2021], [Rahman et al. 2021], [Barbosa et al. 2018], [Yin et al. 2015]
	MK-means	1	[Rahman et al. 2022]
	K-medoids	1	[Combéfis and Schils 2016]
Baseado em hierarquia	Hierarchical Clustering	5	[Silva et al. 2023], [Silva and Silla 2020], [Beh et al. 2016], [Luo and Zeng 2016], [Combéfis and Schils 2016]
Baseado na teoria espectral	Spectral Clustering	2	[Yin et al. 2015], [Rosales-Castro et al. 2016]
Baseado em densidade	Optics	2	[Koivisto and Hellas 2022], [Yin et al. 2015]
	DBSCAN	2	[Koivisto and Hellas 2022], [Yin et al. 2015]
	HDBSCAN	1	[Koivisto and Hellas 2022]
Baseado em afinidade	Affinity Propagation	1	[Rosales-Castro et al. 2016]
Baseado em modelo	Bayesian Gaussian Mixture Models (GMM)	1	[Emerson et al. 2020]
Método Próprio		5	[Effenberger and Pelánek 2021], [Gao et al. 2019], [Head et al. 2017], [Glassman et al. 2015], [Fu et al. 2021]
Ferramenta de cluster		1	[Joyner et al. 2019]

de agrupamento de códigos, que utiliza uma matriz de similaridade construída usando diferentes pesos atribuídos a cinco tipos de similaridades (sequência de tokens, AST, estilo de código, estilo de anotação e recurso estatístico). Já em Head et al. [2017], o algoritmo proposto envolveu a análise das diferenças entre envios incorretos e corretos, para identificar as transformações de código específicas que levam à solução correta.

4.3. Técnicas para visualização dos *clusters* (QP2)

Ao verificar sobre técnicas de visualização para os *clusters*, resultou-se que, dentre as 20 publicações, apenas sete citaram o uso de alguma técnica, como pode ser visto na Tabela 2. Rosales-Castro et al. [2016], Yin et al. [2015], Rahman et al. [2021] e Silva e Silla [2020] aplicaram técnicas de redução de dimensionalidade para que fosse possível visualizar os *clusters* em um espaço bidimensional. No trabalho de Yin et al. [2015], além de aplicar a técnica t-SNE, os autores construíram uma GUI (*Graphical User Interface*) interativa onde os instrutores/professores podem clicar em cada ponto de dados para visualizar o envio do estudante correspondente. A visualização dos *clusters* pode ser visu-

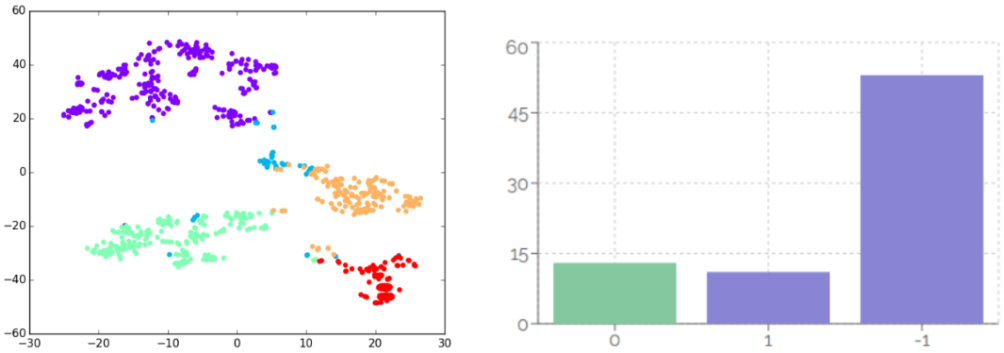
Tabela 2. Técnicas aplicadas para visualização dos *clusters*.

Técnicas de Visualização	Publicações	Descrição
MDS e algoritmo Fruchterman-Reingold	[Rosales-Castro et al. 2016]	Utilizou MDS (*Multidimensional Scaling*) e o algoritmo Fruchterman-Reingold para distribuir as soluções no espaço de visualização.
t-SNE	[Yin et al. 2015]	O t-SNE (*Stochastic Neighbor Embedding*) foi utilizado para reduzir a dimensionalidade dos dados antes de aplicar os algoritmos de *clustering*. Essa técnica ajudou na visualização clara da formação dos *clusters* em espaços de alta dimensionalidade.
	[Rahman et al. 2021]	Utilizou o t-SNE para visualizar resultados de clustering aplicados a métricas de árvore de sintaxe abstrata (AST).
OverCode	[Glassman et al. 2015]	OverCode é um sistema que permite a visualização dos clusters ao exibir uma única solução representativa para cada cluster.
Histogramas	[Koivisto and Hellas 2022]	CodeClusters é um sistema que permite a visualização dos clusters em forma de histograma.
MISTAKEBROWSER	[Head et al. 2017]	O MISTAKEBROWSER é um sistema que possui uma interface onde os professores podem visualizar todos os envios incorretos de um cluster, exibindo fragmentos de código incorretos em vermelho e suas respectivas correções em verde.
Redução de Dimensionalidade (não especificado)	[Silva et al. 2023]	Implementou redução de dimensionalidade para facilitar a visualização de clusters formados em espaços de alta dimensionalidade.

alizada na Figura 3(a), onde se encontram 800 pontos (códigos de estudantes) distribuídos no espaço dimensional, em que cada cor representa um *cluster* diferente [Yin et al. 2015].

Glassman et al. [2015] apresentaram o Overcode, que é um sistema de visualização e exploração de milhares de soluções de programação e, além disso, possui uma interface que tem como foco apresentar informações relevantes para os professores. Apesar de não ser um gráfico de visualização dos *clusters* (algo comum em trabalhos de agrupamento), o Overcode apresenta um código que caracteriza cada *cluster*, além de permitir que os professores filtrem e agrupem soluções com base em critérios diferentes. Em Koivisto e Hellas [2022], o CodeClusters é um sistema projetado para revisão manual eficaz de envios de tarefas de programação e uma das interfaces, também focada nos professores, apresenta histogramas que representam os *clusters*, como pode ser visto na Figura 3(b). Ao clicar em uma das barras que representa um *cluster*, o professor é

direcionado a uma página para escrever *feedbacks*. Já no trabalho de Head et al. [2017], a interface apresenta o código com fragmentos coloridos identificando erros e correções.



(a) (b)

Figura 3. Visualização dos *clusters* com t-SNE em Yin et al. [2015] (a) e Histogramas em Koivisto e Hellas [2022] (b).

4.4. Motivação para o uso de *clusters* (QP3)

Entre as motivações citadas pelos autores para a utilização de métodos de clusterização de soluções de programação semelhantes, os resultados foram categorizados da seguinte forma:

i. **Identificação e correção de erros comuns:** a clusterização é usada para identificar erros comuns entre os estudantes e fornecer *feedback* direcionado para corrigir esses erros [Kawabayashi et al. 2021, Head et al. 2017, Combéfis and Schils 2016, Emerson et al. 2020];
ii. **Geração de *feedback*:** a clusterização é usada para gerar *feedback* personalizado [Joyner et al. 2019, Koivisto and Hellas 2022, Rahman et al. 2021, Head et al. 2017, Beh et al. 2016, Yin et al. 2015, Combéfis and Schils 2016, Fu et al. 2021, Effenberger and Pelánek 2021, Emerson et al. 2020, Gao et al. 2019, Luo and Zeng 2016];
iii. **Decisões pedagógicas:** a clusterização é aplicada para identificar grupos de estudantes com habilidades de programação semelhantes, permitindo que professores tomem decisões pedagógicas específicas, como adaptar o conteúdo, reforçar conceitos ou mudar a metodologia [Silva et al. 2023, Rahman et al. 2022, Silva and Silla 2020];
iv. **Identificação de estratégias de programação e desempenho:** a clusterização é aplicada para identificar estratégias de programação, facilitando a identificação de padrões comuns e a análise do desempenho dos estudantes [Glassman et al. 2015, Fu et al. 2021, Effenberger and Pelánek 2021, Gao et al. 2019, Luo and Zeng 2016];
v. **Aumento da capacidade de revisão do instrutor:** a clusterização é usada para aumentar a capacidade de *feedback* do instrutor ao agrupar soluções semelhantes, facilitando a revisão em grande escala [Barbosa et al. 2018, Rosales-Castro et al. 2016];
vi. **Identificação de estudantes desengajados:** a clusterização é aplicada para detectar alunos desengajados com base no histórico de submissões de código, permitindo intervenções antecipadas [Lokkila et al. 2022].

Como pode ser visto, as motivações para o uso de *clusters*, no contexto de programação, variam entre os trabalhos, em alguns casos são citadas mais de uma motivação [Gao et al. 2019, Effenberger and Pelánek 2021, Fu et al. 2021], que além de identificar estratégias de programação diferentes, têm como objetivo final a geração de *feedback* personalizado conforme os grupos gerados.

Com relação a decisões pedagógicas, em Rahman et al. [2022], os autores encontraram quatro *clusters*, onde: i) *cluster* 1, alta proficiência em programação; ii) *cluster* 2, semelhante ao primeiro, com boa proficiência em programação; iii) *cluster* 3, mais erros do que os dois primeiros *clusters*; iv) *cluster* 4, baixa precisão e menos códigos aceitos, o que demonstrou um esforço menor que os demais. Esses achados foram importantes para a adaptação de estratégias e intervenções de ensino para atender a necessidades específicas. Já no contexto de identificação e correção de erros comuns, em Kawabayashi et al. [2021], a partir da clusterização, foi possível identificar os padrões de erros mais frequentes em códigos de programação, como por exemplo, nomes de variáveis incorretas e tamanho de *array* incorreto.

Em suma, as possibilidades de apoio à aprendizagem de programação ao utilizar clusterização de soluções semelhantes são diversas, desde a tomada de decisões pedagógicas, que podem auxiliar nas dificuldades dos estudantes, até a identificação de alunos desengajados, permitindo interferir previamente para tentar evitar a evasão/reprovação nas disciplinas.

4.5. Consolidação dos resultados

A Tabela 3 apresenta uma visão geral das publicações identificadas neste MSL, contendo as informações sobre o contexto (disciplina), a linguagem de programação dos códigos que formaram as bases de dados para os experimentos de clusterização e qual o método abordado em cada artigo.

Como pode ser visto, apenas em dez publicações os autores informaram a disciplina a qual se tratavam os códigos-fonte usados nos experimentos de clusterização, sendo que em nove o contexto foi Introdução à Programação e em somente uma foi Algoritmos e Estrutura de Dados. Com relação à linguagem, Python foi a mais citada, em seis das publicações, e apenas três artigos não informaram qual era a linguagem de programação com a qual estavam trabalhando.

Todos os artigos citaram qual método foi aplicado para gerar os *clusters* e o algoritmo K-means se destacou, sendo utilizado em seis das publicações. A última coluna da Tabela 3, intitulada "T.V.", refere-se ao uso de alguma técnica para visualização dos *clusters*, e, como pode ser visto, isso foi identificado em somente sete publicações.

Tabela 3. Visão geral das publicações identificadas com o MSL.

ID	Artigo	Contexto	Linguagem	Método	T.V.
1	[Joyner et al. 2019]	Introdução à Programação	Python	Ferramenta de cluster	Não
2	[Lokkila et al. 2022]	Introdução à Programação	Não relatada	K-means	Não
3	[Silva et al. 2023]	Não relatado	C	Hierárquico	Sim

4	[Koivisto and Hellas 2022]	Não relatado	Java	K-means, DBSCAN, HDBSCAN, OPTICS	Sim
5	[Rahman et al. 2022]	Algoritmos e Estrutura de Dados	Não relatada	MK-means	Não
6	[Kawabayashi et al. 2021]	Não relatado	C++	K-means	Não
7	[Fu et al. 2021]	Introdução à Programação	Java e C#	Método próprio	Não
8	[Effenberger and Pelánek 2021]	Introdução à Programação	Python	Método próprio	Não
9	[Silva and Silla 2020]	Não relatado	C	Hierárquico	Não
10	[Rahman et al. 2021]	Não relatado	Não relatada	K-means	Sim
11	[Emerson et al. 2020]	Introdução à Programação	Blocos	Bayesian Gaussian Mixture Models	Não
12	[Gao et al. 2019]	Não relatado	C	Método próprio	Não
13	[Barbosa et al. 2018]	Não relatado	Python	K-means	Não
14	[Head et al. 2017]	Não relatado	Python	Método próprio	Sim
15	[Beh et al. 2016]	Introdução à Programação	Java	Hierárquico	Não
16	[Yin et al. 2015]	Introdução à Programação	Ruby	K-means, Spectral, DBSCAN, OPTICS	Sim
17	[Luo and Zeng 2016]	Não relatado	Java	Hierárquico	Não
18	[Rosales-Castro et al. 2016]	Não relatado	Python	Affinity Propagation, Spectral	Sim
19	[Glassman et al. 2015]	Introdução à Programação	Python	Método próprio	Sim
20	[Combéfis and Schils 2016]	Não relatado	C#	K-medoids e Hierárquico	Não

5. Considerações finais

Este Mapeamento Sistemático da Literatura (MSL) teve como objetivo identificar os trabalhos existentes na literatura sobre o uso de clusterização aplicado a soluções de exercícios de programação. A partir dos 20 artigos selecionados, foi possível detectar quais os métodos utilizados pelos pesquisadores da área, o que motiva o uso dessa abordagem e se são exploradas técnicas para a visualização dos *clusters*.

A QP1 tem como foco identificar os métodos de clusterização aplicados. A partir da leitura dos artigos, foram encontradas as seguintes abordagens: K-means e variações deste mesmo algoritmo; hierárquico; espectral; OPTICS; DBSCAN; HDBSCAN; *Affinity Propagation*; e, *Bayesian Gaussian Mixture Models* (GMM). Além disso, alguns trabalhos usaram ferramentas prontas que permitem a geração de *cluster* e outros apresentaram seus próprios métodos. Com a QP2 buscou-se entender se os trabalhos utilizaram alguma técnica para visualizar os *clusters*, e apenas em sete artigos isso foi identificado. Alguns trabalhos somente aplicaram uma técnica de redução de dimensionalidade para que fosse possível visualizar os *clusters* graficamente, enquanto em outros artigos, os autores apresentaram sistemas com interfaces focadas em apresentar visualmente os *clusters* para professores da área.

A última questão de pesquisa (QP3) procurou entender quais as motivações dos pesquisadores que aplicaram clusterização para agrupar soluções de exercícios de programação. A identificação de estratégias de programação e desempenho foi uma das motivações mais citadas pelos autores, assim como a geração de *feedback* adequado e personalizado. Outros trabalhos apontaram o uso de *clusters* para a tomada de decisões pedagógicas, como adaptação de conteúdo e mudança na metodologia. Alguns artigos usaram os métodos de agrupamento para aumentar a capacidade de revisão em grande escala e para a identificação de erros comuns que ocorrem entre os estudantes. Um dos autores aplicou a abordagem para a detecção de estudantes desengajados, o que pode ajudar em intervenções antecipadas.

A partir deste mapeamento foi possível notar a diversidade de métodos de clusterização empregados para agrupar soluções de exercícios de programação, bem como a quantidade de pesquisas na área que buscam entender essas estratégias e, assim, auxiliar a aprimorar o processo de aprendizagem em programação. Os métodos que variam desde K-means até técnicas mais modernas como o *Spectral Clustering*, mostram que a complexidade e a variedade das estratégias de programação exigem ferramentas robustas para análise e agrupamento. Com tais abordagens, é possível melhorar a qualidade do *feedback* fornecido aos estudantes e entender quais são as decisões pedagógicas necessárias para tratar as necessidades dos estudantes.

Este trabalho traz como principais contribuições: i) base para pesquisadores da área; ii) possibilidade de novos estudos, como por exemplo, entender quais os melhores métodos a serem utilizados; iii) pode fornecer *ingishts* sobre outros campos em que técnicas de aprendizagem de máquina podem ser aplicadas no contexto da aprendizagem de programação; iv) informações sobre técnicas que podem ser utilizadas para melhorar tanto a experiência dos professores quanto dos estudantes em ambientes de aprendizagem em programação; e, v) identificação de lacunas, como a baixa quantidade de trabalhos abordando técnicas de visualização dos *clusters*, o que pode ser importante para ajudar os professores a entender sobre os dados dos grupos de estudantes.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD 2024/2025.

O presente trabalho também é decorrente do projeto de Pesquisa e Desenvolvimento (P&D) 001/2020, firmado entre a Fundação da Universidade do Amazonas e FAEPI, que conta com financiamento da Samsung, usando recursos da Lei de Informática para a Amazônia Ocidental (Lei Federal nº 8.387/1991), estando sua divulgação de acordo com o previsto no artigo 39.º do Decreto nº 10.521/2020.

Referências

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

Barbosa, A. d. A., Costa, E. d. B., and Brito, P. H. (2018). Adaptive clustering of codes for assessment in introductory programming courses. In *Intelligent Tutoring Systems:*

14th International Conference, ITS 2018, Montreal, QC, Canada, June 11–15, 2018, Proceedings 14, pages 13–22. Springer.

Barbosa, A. d. A., de Barros Costa, E., and Brito, P. H. (2023). Juízes online são suficientes ou precisamos de um var? In *Anais do III Simpósio Brasileiro de Educação em Computação*, pages 386–394. SBC.

Beh, M. Y., Gottipatti, S., LO, D., and Shankararaman, V. (2016). Semi-automated tool for providing effective feedback on programming assignments.

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Combéfis, S. and Schils, A. (2016). Automatic programming error class identification with code plagiarism-based clustering. In *Proceedings of the 2nd International Code Hunt Workshop on Educational Software Engineering*, pages 1–6.

Effenberger, T. and Pelánek, R. (2021). Interpretable clustering of students' solutions in introductory programming. In *International Conference on Artificial Intelligence in Education*, pages 101–112. Springer.

Emerson, A., Smith, A., Rodriguez, F. J., Wiebe, E. N., Mott, B. W., Boyer, K. E., and Lester, J. C. (2020). Cluster-based analysis of novice coding misconceptions in block-based programming. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 825–831.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fu, Y., Osei-Owusu, J., Astorga, A., Zhao, Z. N., Zhang, W., and Xie, T. (2021). Pacon: a symbolic analysis approach for tactic-oriented clustering of programming submissions. In *Proceedings of the 2021 ACM SIGPLAN International Symposium on SPLASH-E*, pages 32–42.

Gao, L., Wan, B., Fang, C., Li, Y., and Chen, C. (2019). Automatic clustering of different solutions to programming assignments in computing education. In *Proceedings of the ACM Conference on Global Computing Education*, pages 164–170.

Glassman, E. L., Scott, J., Singh, R., Guo, P. J., and Miller, R. C. (2015). Overcode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 22(2):1–35.

Head, A., Glassman, E., Soares, G., Suzuki, R., Figueredo, L., D'Antoni, L., and Hartmann, B. (2017). Writing reusable code feedback at scale with mixed-initiative program synthesis. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pages 89–98.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Joyner, D., Arrison, R., Ruksana, M., Salguero, E., Wang, Z., Wellington, B., and Yin, K. (2019). From clusters to content: Using code clustering for course improvement. In

Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pages 780–786.

Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

Kawabayashi, S., Rahman, M. M., and Watanobe, Y. (2021). A model for identifying frequent errors in incorrect solutions. In *2021 10th International Conference on Educational and Information Technology (ICEIT)*, pages 258–263. IEEE.

Kitchenham, B., Madeyski, L., and Budgen, D. (2022). Segress: Software engineering guidelines for reporting secondary studies. *IEEE Transactions on Software Engineering*, 49(3):1273–1298.

Koivisto, T. and Hellas, A. (2022). Evaluating codeclusters for effectively providing feedback on code submissions. In *2022 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Lokkila, E., Christopoulos, A., and Laakso, M.-J. (2022). A clustering method to detect disengaged students from their code submission history. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, pages 228–234.

Luo, L. and Zeng, Q. (2016). Solminer: mining distinct solutions in programs. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 481–490.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Pessoa, M., Lima, M., Pires, F., Haydar, G., Melo, R., Rodrigues, L., Oliveira, D., Oliveira, E., Galvão, L., Gadelha, B., et al. (2023). A journey to identify users' classification strategies to customize game-based and gamified learning environments. *IEEE Transactions on Learning Technologies*.

Rahman, M. M., Watanobe, Y., Matsumoto, T., Kiran, R. U., and Nakamura, K. (2022). Educational data mining to support programming learning using problem-solving data. *IEEE Access*, 10:26186–26202.

Rahman, M. M., Watanobe, Y., Rage, U. K., and Nakamura, K. (2021). A novel rule-based online judge recommender system to promote computer programming education. In *Advances and Trends in Artificial Intelligence. From Theory to Practice: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part II 34*, pages 15–27. Springer.

Rosales-Castro, L. F., Chaparro-Gutiérrez, L. A., Cruz-Salinas, A. F., Restrepo-Calle, F., Camargo, J., and González, F. A. (2016). An interactive tool to support student assessment in programming assignments. In *Advances in Artificial Intelligence-IBERAMIA 2016: 15th Ibero-American Conference on AI, San José, Costa Rica, November 23-25, 2016, Proceedings 15*, pages 404–414. Springer.

Silva, D. B., Carvalho, D. R., and Silla, C. N. (2023). A clustering-based computational model to group students with similar programming skills from automatic source code analysis using novel features. *IEEE Transactions on Learning Technologies*.

Silva, D. B. and Silla, C. N. (2020). Evaluation of students programming skills on a computer programming course with a hierarchical clustering algorithm. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17:395–416.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Wasik, S., Antczak, M., Badura, J., Laskowski, A., and Sternal, T. (2018). A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)*, 51(1):1–34.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of data science*, 2:165–193.

Yin, H., Moghadam, J., and Fox, A. (2015). Clustering student programming assignments to multiply instructor leverage. In *Proceedings of the second (2015) ACM conference on learning@ scale*, pages 367–372.