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

Descoberta de padrões sequenciais de aprendizagem em um ambiente voltado ao ensino de algoritmos

Djefferson Maranhão[1], **Paulo Victor Borges**[1], **Carlos de Salles Soares Neto**[1]

[1]Departamento de Informática – Universidade Federal do Maranhão (UFMA)
São Luis – MA – Brazil

djefferson.maranhao@gmail.com, pvbo.lima@discente.ufma.br,

carlos.salles@ufma.br

Abstract. *This paper qualitatively analyzes the use of Sequential Pattern Mining to discover learning patterns in an environment dedicated to teaching algorithms. The question to be faced is how to improve the students' learning experience through the investigation of navigation patterns by the computational problems available in this environment. The proposed approach is able to reveal to the teacher how the student's learning experience is going, so that he can act to prevent a possible demotivation towards the platform. The main contribution of this work was to show that this approach can generate valuable information for the teacher, such as the need to review a problem or to provide more examples for students*

Resumo. *O presente trabalho analisa de forma qualitativa a utilização da Mineração de Padrões Sequenciais para a descoberta de padrões de aprendizagem em um ambiente voltado ao ensino de algoritmos. A questão a ser enfrentada é como melhorar a experiência de aprendizagem dos alunos por meio da investigação dos padrões de navegação pelos problemas computacionais disponibilizados nesse ambiente. A abordagem proposta é capaz de revelar ao professor como está sendo a experiência de aprendizagem do aluno, de modo que ele possa atuar na prevenção de uma eventual desmotivação em relação à plataforma. A principal contribuição deste trabalho foi mostrar que essa abordagem pode gerar informações valiosas para o professor, como a necessidade de revisar um problema ou de fornecer mais exemplos aos alunos.*

1. Introdução

Ensinar programação para quem está iniciando no mundo da computação é uma tarefa desafiadora, pois muitos dos conceitos e habilidades envolvidos podem parecer demasiadamente complexos à primeira vista. É o caso de algoritmos, que representam uma parte essencial da computação, mas que podem facilmente causar confusão se não forem ensinados da maneira correta.

Uma forma de simplificar o ensino de programação é mediante a introdução gradual de novos conceitos. Começa-se ensinando o conceito de declaração e inicialização de variáveis; depois, explica-se o funcionamento dos operadores aritméticos, de comparação e lógicos. Em seguida, apresenta-se o conceito de estruturas de decisão, o qual é necessário para a compreensão de laços de repetição.

Essa maneira de ensinar programação é marcada por etapas bem definidas, que ocorrem em momentos distintos. Isso permite explorar o ensino de programação sob a ótica da análise temporal do aprendizado, com o propósito de: detectar transições entre eventos de aprendizagem; identificar variações no processo de aprendizagem; explicar variações nos resultados da aprendizagem; e acelerar o surgimento de novas questões [Molenaar and Wise 2022].

Em geral, duas propriedades temporais da aprendizagem são investigadas: a passagem do tempo e a ordem no tempo. A passagem do tempo diz respeito a quando; com que frequência; ou por quanto tempo ocorrem os eventos de aprendizagem nos quais se está interessado. Uma limitação dessa propriedade é que ela pode esconder ocorrências antes e depois dos eventos de interesse, ou seja, o contexto em que o evento ocorre.

Por outro lado, a ordem no tempo aborda essa limitação concentrando-se no arranjo relativo dos eventos de aprendizagem. Recentemente, essa propriedade foi explorada no exame das relações entre o raciocínio clínico e a eficiência do diagnóstico realizado por estudantes de medicina [Zheng et al. 2022]; e para revelar fases e estratégias do processo de resolução de problemas por estudantes [Liu and Israel 2022].

A mineração de padrões sequenciais (MPS) é um processo poderoso para a descoberta de padrões ocultos em eventos ordenados no tempo. Em ambientes voltados ao ensino de algoritmos, por exemplo, padrões sequenciais frequentes podem indicar comportamentos comuns entre os alunos. Esses padrões de comportamento revelam como os alunos navegam entre os problemas computacionais e ajudam a melhorar a experiência de aprendizagem.

O objetivo deste trabalho é analisar de forma qualitativa a utilização da MPS para descoberta de padrões ocultos em dados de uso de um ambiente voltado ao ensino de algoritmos. Essa abordagem é capaz de revelar ao professor como está sendo a experiência de aprendizagem do aluno, permitindo que ele possa atuar de maneira a prevenir uma eventual desmotivação com o uso da plataforma.

O artigo está organizado como segue: a Seção 2 apresenta a fundamentação teórica acerca da mineração de padrões sequenciais e suas terminologias; a Seção 3 apresenta os trabalhos que se relacionam a esta pesquisa; a Seção 4 descreve a metodologia de análise proposta; e a Seção 5 apresenta a conclusão do trabalho.

2. Mineração de Padrões Sequenciais (MPS)

A MPS consiste em encontrar subsequências interessantes em um conjunto de dados de sequências [Agrawal and Srikant 1995]. O interesse de uma subsequência pode ser definido de várias maneiras, como sua frequência ou seu comprimento. Uma sequência pode ser representada por $\{i_1, i_2, \cdots, i_n\}$, em que i_j é um conjunto não vazio de itens (*itemset*). No contexto de algoritmos, um item representa a ação realizada por um aluno ao submeter a resolução de um problema no ambiente de ensino.

A Tabela 1 apresenta um conjunto de sequências para alunos hipotéticos. A sequência do aluno 1 possui duas interações. O primeiro *itemset* contém dois itens, a e b, indicando que o aluno interagiu com os objetos de aprendizagem a e b na sessão 1, enquanto a sessão 2 indica que o aluno interagiu com o problema c, depois na sessão 3 com o problema d e, finalmente, na sessão 4 com o problema e. A ordem dos itens em

um *itemset* não é relevante. Todas as sessões do aluno formam uma sequência.

Tabela 1. Exemplo de Banco de Sequências.

Id do Aluno	Sequência
1	$<\{a,b\},\{c\},\{d\},\{e\}>$
2	$<\{a,c\},\{e,f\}>$
3	$<\{b\},\{c,d,e\},\{f,b\},\{c\}>$
4	$<\{b,c\},\{d\},\{f\},\{g\}>$

Uma sequência $\{i_1, i_2, \cdots, i_n\}$ contém uma subsequência $\{p_1, p_2, \cdots, p_m\}$ se houver números inteiros $k_1 < k_2 < \cdots < k_n$ tais que $p_1 \subseteq i_{k_1}, p_2 \subseteq i_{k_2}, p_m \subseteq i_{k_n}$. Por exemplo, as sequências 1 e 3 na Tabela 1 contêm a subsequência $<\{b\},\{c\}>$, mas as sequências 2 e 4 não.

Muitos algoritmos permitem que se defina o *gap* máximo para reduzir padrões ruidosos (padrões identificados como frequentes devido a erros aleatórios) e limitar o número de padrões retornados [Srikant and Agrawal 1996][Zaki 2000]. Por exemplo, se o intervalo máximo for 1, o algoritmo inferirá que a sequência do cliente 1 não contém $<\{b\},\{e\}>$ porque o intervalo entre b e e nessa sequência é 3. Os ruídos devem ser removidos porque eles podem não fornecer informações confiáveis sobre os processos que geraram as sequências.

A sequência de um aluno suporta uma subsequência se ela estiver contida na sequência do aluno. O valor de suporte de uma subsequência é definido como a proporção de sequências que contém essa subsequência [Agrawal and Srikant 1995]. Se o valor de suporte de uma subsequência não for menor que um limite pré-especificado (chamado de suporte mínimo), essa subsequência é um padrão sequencial frequente.

Suponha, como exemplo, que o suporte mínimo é definido como 0,5, então todas as sequências com suporte mínimo acima desse limiar serão consideradas frequentes. Na Tabela 1, os valores de suporte de $<\{b\},\{c\}>$, $<\{b\},\{d\}>$, $<\{c\},\{d\}>$ são, respectivamente, 0,5, 0,75 e 0,5, portanto, essas subsequências são consideradas frequentes. Em contraste, $<\{a\},\{c\}>$ não é um padrão frequente porque seu suporte é $0,25$, o que é menor que $0,5$.

Da mesma forma que no uso de um intervalo máximo, selecionar o suporte mínimo apropriado ajuda a reduzir padrões ruidosos, pois quanto mais sequências contêm um padrão específico, menos provável é que esse padrão ocorra devido a um erro aleatório. Outro valor frequentemente utilizado é o valor da instância [Lo et al. 2008] que se refere ao número de ocorrências de um padrão sequencial em uma sequência. Por exemplo, se não considerarmos o intervalo máximo, os valores de instância de $<\{b\},\{c\}>$ nas sequências 1 a 4 são, respectivamente, 1, 0, 2 e 1.

2.1. Métodos para Mineração de Padrões Sequenciais

Vários algoritmos têm sido aplicados a dados educacionais, como GSP (*Generalized Sequential Patterns*) [Srikant and Agrawal 1996]; SPADE [Zaki 2000]; FreeSpan [Han et al. 2000], PrefixSpan [Han et al. 2001]; SPAM (*Sequential Pattern Mining*) [Ayres et al. 2002] e LAPIN (*Last Position Induction*) [Yang et al. 2007]. Esses algoritmos podem ser divididos em duas abordagens:

- Geração de Candidatos: essa abordagem se baseia na execução de múltiplos passos sobre os dados. Em cada passo, começa-se com um conjunto semente de um número grande de sequências, chamadas de sequências candidatas. O suporte para essas sequências candidatas é computado durante a passada sobre os dados. Ao final do passo, são determinadas as maiores sequências candidatas. Tais sequências compõem a semente para o próximo passo. Como exemplos, podem-se citar os algoritmos GSP e SPADE;
- Crescimento de Padrões: essa abordagem usa a estratégia de dividir para conquistar, começando com um conjunto de padrões frequentes de tamanho 1. Em seguida, deriva, para cada padrão p, um banco de dados projetado de p e o explora recursivamente. Como o conjunto de dados é decomposto progressivamente em um conjunto de bancos de dados muito menores, o método de crescimento de padrões reduz o espaço de pesquisa e leva a alta eficiência e escalabilidade. Como exemplos, podem-se citar os algoritmos FreeSpan, PrefixSpan e LAPIN.

3. Trabalhos Relacionados

Na área educacional, a mineração de padrões sequenciais tem sido empregada em pesquisas com propósitos variados, que incluem a descoberta de comportamentos de aprendizagem; o enriquecimento de teorias educacionais; e a filtragem de recursos de aprendizagem para a construção de sistemas de recomendação. Inserido nesse contexto, o presente trabalho busca contribuir com o estado da arte investigando o uso dessa abordagem no contexto do ensino de algoritmos.

Os dados do processo de aprendizagem, como *logs* de eventos, registram informações detalhadas sobre as interações dos alunos com os ambientes de aprendizagem, colegas e instrutores. Nesse contexto, padrões sequenciais frequentes podem indicar comportamentos comuns entre os alunos [Zhou et al. 2010]. Esses padrões de comportamento podem revelar como os alunos navegam em suas atividades dentro de um ambiente de aprendizagem e informar como atualizar melhor a experiência de aprendizagem [Mirzaei and Sahebi 2019].

Em [Kang et al. 2017], os autores aplicaram o algoritmo cSPADE aos *logs* de *Alien Rescue*, um jogo sério para ensinar habilidades científicas de resolução de problemas a alunos do ensino médio. O estudo teve como objetivo analisar como os padrões sequenciais de interação podem diferir ao longo de vários dias de uso do jogo educacional. Eles observaram como os padrões sequenciais nos primeiros dias representavam comportamentos de exploração, enquanto os padrões sequenciais nos dias restantes indicavam comportamentos científicos de resolução de problemas.

Alguns estudos mostram como a mineração de padrões sequenciais pode ser utilizada para investigar trajetórias de grupos com desempenhos acadêmicos diferentes para tentar identificar padrões frequentes no grupo de alto desempenho, mas raros no grupo de baixo desempenho. Por exemplo, [Slim et al. 2016] aplicaram a MPS em sequências de matrículas em cursos de graduação em engenharia elétrica. O resultado mostrou que os alunos que se formaram com uma média alta seguiram um padrão de matrícula distinto daqueles que obtiveram uma média mais baixa.

4. Metodologia

Esta seção apresenta o processo de análise de dados adotado neste trabalho. Na Seção 4.1, apresenta-se o ambiente no qual os dados foram coletados. A Seção 4.2 explora brevemente o conjunto de dados, para dar um maior entendimento ao leitor. Em seguida, na Seção 4.3, são discutidas as atividades realizadas no contexto da preparação e transformação dos dados relacionais em sequenciais. A Seção 4.4 refere-se à técnica de mineração de padrões sequenciais e, por último, a Seção 4.5 discute os resultados obtidos.

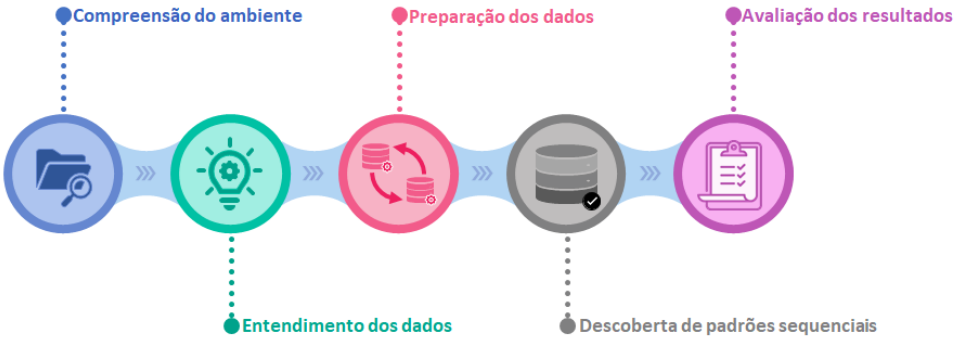

Figura 1. Metodologia de análise.

4.1. Compreensão do ambiente

Esta etapa consiste em entender um pouco da dinâmica do ambiente de aprendizagem de programação no qual os dados em análise foram produzidos. A plataforma de ensino em questão é utilizada na disciplina de Algoritmos I do Curso de Ciência da Computação da Universidade Federal do Maranhão. Essa disciplina foi planejada de modo a ser realizada no período de um semestre, contemplando os tópicos: variáveis e atribuição; comandos condicionais; laços de repetição; vetores e listas. No período de 03 de março de 2021 a 16 de dezembro de 2022, quando os dados foram produzidos, a disciplina foi conduzida sempre pelo mesmo professor.

A cada semestre, uma nova turma é cadastrada na plataforma de ensino. Inicialmente, quando o aluno entra na plataforma pela primeira vez, ele tem acesso apenas ao conteúdo teórico e aos problemas relacionados ao tópico de variáveis e atribuição. Conforme os assuntos vão sendo abordados em sala de aula, os demais tópicos vão sendo liberados sequencialmente, em média, a cada 6 semanas. A plataforma permite que os alunos acessem o conteúdo teórico e naveguem livremente pelas questões de cada tópico, não oferecendo qualquer tipo de suporte adicional à navegação.

Com o avanço das aulas ao longo do semestre, o professor vai sugerindo a resolução dos problemas cadastrados na plataforma, que conta com um sistema de juiz *online* capaz de compilar os códigos submetidos pelos alunos e executá-los contra baterias de teste previamente definidas, avaliando se os códigos produzem os resultados esperados (sucesso); se falham ao compilar (erro de compilação); se são incapazes de gerar os resultados esperados no tempo previsto (tempo limite excedido); ou se não conseguem gerar os resultados aguardados (erro de execução).

4.2. Entendimento dos dados

Esta etapa consiste em se entender quais dados se possui, onde eles podem ser obtidos, o que há no conjunto de dados e se eles possuem a qualidade necessária. A coleta dos dados foi realizada na base de dados da plataforma de ensino, a partir do cruzamento entre as tabelas de submissão e de questão, resultando em um conjunto de dados composto pelas seguintes colunas: $usuario_id$, $questao_id$, $conceito_id$, $turma_id$, $tipo_resultado_id$, $tipo_linguagem_id$, $codigo$, $data_criacao$, $tempo_inicial$, $tempo_final$ e $resultado$, conforme apresentado na Figura 2.

id	usuario_id	questao_id	...	tipo_resultado_id	tipo_linguagem_id	codigo	tempo_inicial	created_at	tempo_final	resultado
45	40	3	...	3	5	print("Olá Mundo!)		2021-03-03 17:21:13		File "codigo.py", line 1 print("Ol\xe1 Mundo!) ^ SyntaxError: EOL while scanning string literal
46	13	3	...	1	5	print("Hello World!")	1614792076	2021-03-03 17:21:16	1614792076	Hello World!
47	31	3	...	1	5	print ("Hello World!")	1614792093	2021-03-03 17:21:33	1614792093	Hello World!
...	...	...	...	...	...	...	...	...	...	...
10972	267	25	...	2	5	n= int(input()) soma=0 for i in range(0,n): soma+=i print(soma)	1671033383	2022-12-14 15:56:24	1671033383	10 6 21 3
11000	131	3	...	3	5	print("Hello World!)		2022-12-14 23:51:56		File "codigo.py", line 1 print("Hello World!) ^ SyntaxError: EOL while scanning string literal
11001	131	3	...	1	5	print("Hello World!")	1671061984	2022-12-14 23:53:04	1671061984	Hello World!

9562 rows x 12 columns

Figura 2. Conjunto de dados a ser analisado nas etapas seguintes.

O conjunto de dados contém um total de 9.562 (nove mil, quinhentos e sessenta e duas) submissões realizadas por 229 alunos de 6 turmas (semestres) diferentes. Todos os códigos foram escritos na linguagem Python. Conforme enumerado na Tabela 2, a plataforma disponibilizou um total de 22 (vinte e dois) problemas, sendo 6 referentes ao tópico de variáveis e atribuição; 9 de comandos condicionais; 4 de laços de repetição; e 3 de vetores e listas.

Tabela 2. Distribuição das questões por tópico.

Tópico	Identificadores
Variáveis e Atribuição	$3, 4, 5, 6, 7, 8$
Comandos Condicionais	$10, 11, 12, 13, 14, 15, 18, 19, 30$
Laços de Repetição	$16, 17, 26, 31$
Vetores e Listas	$25, 27, 29$

Cada submissão na plataforma pode resultar em sucesso, erro de compilação, tempo limite excedido ou erro de execução. No conjunto de dados, estão presentes 3.169 soluções bem-sucedidas; 4.651 soluções com erros de compilação; e 1.742 soluções que resultaram em erros de execução. Os tipos de resultado estão distribuídos pelas questões conforme representado na Figura 2.

O número médio de interações por aluno é de 41,75, sendo o desvio padrão de cerca de 32,98. O mínimo de interações por aluno foi de apenas uma interação, enquanto o máximo foi de 184 submissões para apenas um aluno. O número médio de interações por questão é de 434,63, sendo o desvio padrão de aproximadamente 255,37. A questão com a qual os alunos menos interagiram obteve um total de 44 submissões, enquanto a que teve mais interações obteve um total de 878 submissões. Outras medidas-resumo do conjunto de dados estão descritas nas Tabelas 2 e 3.

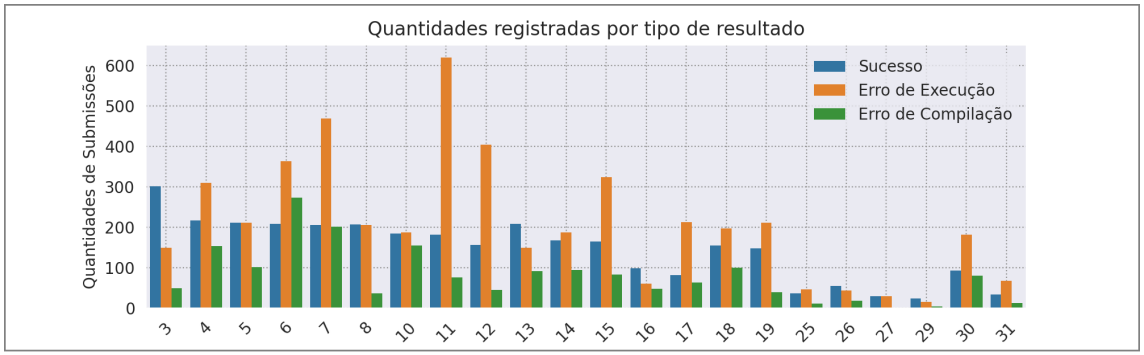

Figura 3. Distribuição dos tipos de resultado por questão.

Medida-resumo	Valor
Média	41,75
Desvio Padrão	32,98
Mínimo	1
1o. Quartil (25%)	20
2o. Quartil (50%	32
3o. Quartil (75%)	56
Máximo	184

Tabela 3. Medidas-resumo da distribuição de interações por usuário.

Medida-resumo	Valor
Média	434,63
Desvio Padrão	255,37
Mínimo	44
1o. Quartil (25%)	243,25
2o. Quartil (50%	450
3o. Quartil (75%)	561
Máximo	878

Tabela 4. Medidas-resumo da distribuição de interações por questão.

4.3. Preparação dos Dados

Esta etapa é responsável, basicamente, pelas atividades de seleção, limpeza, enriquecimento e transformação dos dados. Os valores discrepantes, *outliers*, foram identificados usando o método do intervalo interquartílico ($IQR = Q_3 - Q_1 = 56 - 20 = 36$). Quaisquer observações menores que $Q_1 - 1,5 \times IQR = 20 - 1,5 \times 36 = -32,5$ ou maiores que e $Q_3 + 1,5 \times IQR = 20 + 1,5 \times 36 = 110$ são consideradas *outliers*. Nesse processo, foram excluídos da análise 10 usuários e 1.393 submissões.

Em seguida, os registros de submissão em formato relacional foram transformados em uma estrutura de lista. Essa etapa consistiu em ordenar cronologicamente os registros de submissão de cada usuário de modo a se obter sua sequência de interações com a plataforma de ensino. Após isso, essa representação em lista ainda foi transformada em um arquivo de texto no formato aceito pela biblioteca SPMF [Fournier-Viger et al. 2014], a qual é utilizada neste trabalho.

4.4. Descoberta de Padrões Sequenciais

Para a etapa de descoberta de padrões, foram utilizadas duas abordagens distintas. A primeira consistiu na construção de um grafo, em que os nós representam as questões juntamente com a quantidade de tentativas por um mesmo aluno. As arestas representam o número de vezes em que os alunos transitaram de uma determinada questão para outra. Como ponto de partida, adicionou-se um nó de início. Assim, após se cadastrar, se um determinado usuário respondeu a primeira questão e, em seguida, partiu para a terceira questão, considera-se ter havido uma sequência *início* $\rightarrow 1 \rightarrow 3$.

Essa estrutura em formato de grafo permite que se possa visualizar mais facilmente os dados e explorar padrões porventura neles existentes. Na Figura 4, por exemplo, mostra-se a poda das arestas do grafo com base em um *threshold* previamente estabelecido. Para um *threshold* $= 30$, surgem padrões relevantes que serão discutidos na próxima seção.

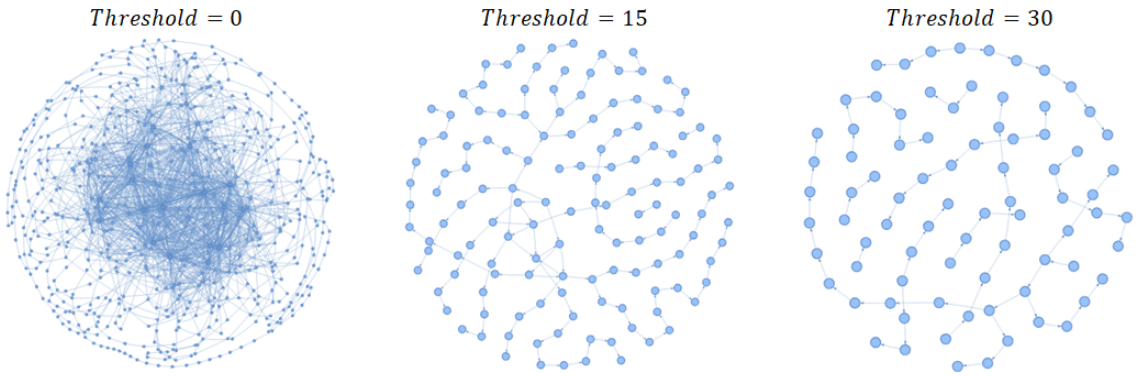

Figura 4. Representação da abordagem de poda em grafo

A outra abordagem consistiu na utilização da ferramenta SPMF [Fournier-Viger et al. 2014], uma biblioteca de código fonte aberto composta por diversos algoritmos relacionados à tarefa de descoberta de padrões sequenciais em *datasets*. Para essa tarefa, foram escolhidos os algoritmos GSP, SPADE, FreeSpan, SPAM e LAPIN. Esses algoritmos foram então executados contra o *dataset* considerando-se valores de suporte mínimo variando no intervalo de $0,1$ a 1. O número de padrões detectados, o tempo de processamento e a memória consumida em KB estão representados, respectivamente, nas Figuras 5, 6 e 7.

4.5. Avaliação dos Resultados

Nesta etapa são discutidos os resultados obtidos na seção anterior. A seção está dividida em duas partes, uma aborda os resultados obtidos com a poda do grafo e a outra avalia os resultados obtidos com os algoritmos de mineração. As duas abordagens citadas na seção anterior mostram-se capazes de gerar *insights* valiosos para o professor da disciplina de Algoritmos I.

Conforme ilustra a Figura 8, a sequência obtida a partir do método de poda do grafo, para um *threshold* $= 30$, mostra a dificuldade dos alunos com as questões 6 e 7 do tópico variáveis e atribuição, que normalmente exigem mais tentativas. Esse padrão é consistente porque essas duas questões apresentam um diferencial em relação às demais questões do mesmo tópico, que é a exigência de formatação da saída com um certo número de casas decimais. A detecção desse tipo de padrão é fundamental para que se consiga reverter um eventual processo de desmotivação do aluno com relação ambiente de ensino, disparado pelos sucessivos erros em uma mesma questão. Por isso, é fundamental que o professor revise os enunciados desses dois problemas, ou forneça mais exemplos de como utilizar a formatação de casas decimais na linguagem Python.

Além disso, esse padrão também mostra que os alunos costumam responder as questões do tópico variáveis e atribuição conforme a ordem em que as questões são disponibilizadas na interface gráfica, que é 3, 4, 5, 6, 7 e 8. Em seguida, os alunos partem para

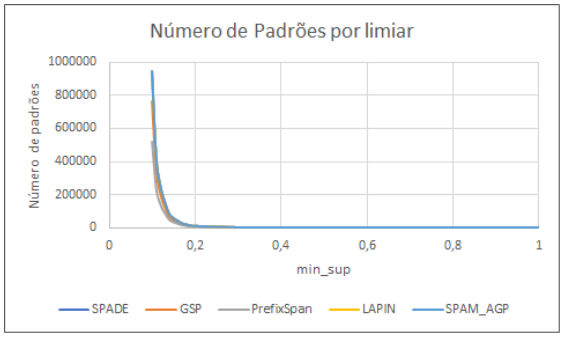

Figura 5. Número de Padrões vs. suporte mínimo

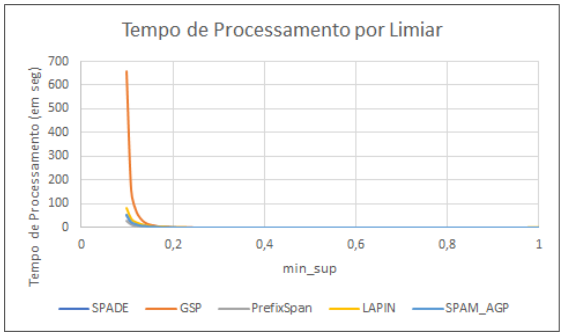

Figura 6. Tempo de Processamento vs. suporte mínimo

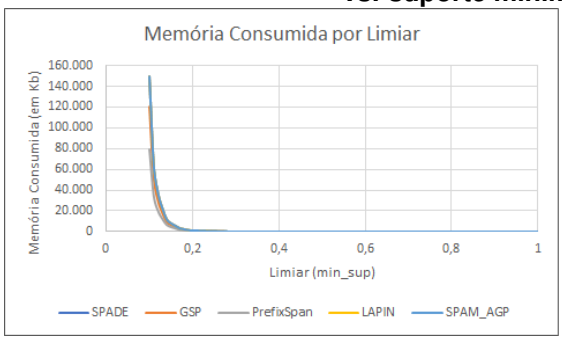

Figura 7. Memória Consumida vs. suporte mínimo

resolver as questões de comandos condicionais. Pode-se verificar que, nesse momento, começa a haver uma diminuição no número de interações com a plataforma, muito provavelmente, por conta do nível de exigência das questões 11 e 12, que envolvem não apenas comandos condicionais, mas também conceitos matemáticos de sistemas lineares.

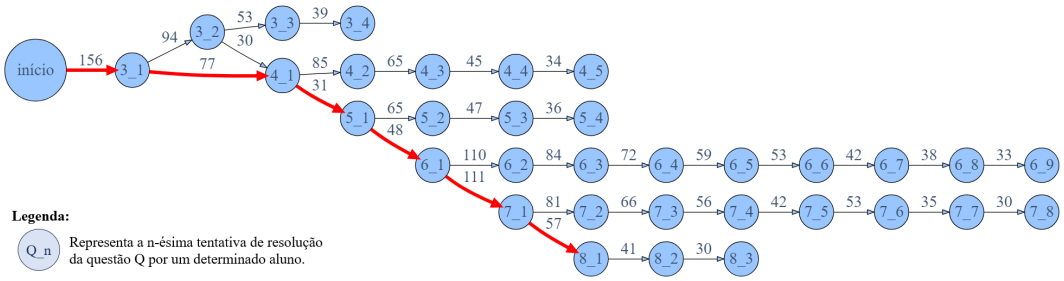

Figura 8. Sequência obtida a partir da poda do grafo.

Na abordagem utilizando algoritmos de mineração, verificou-se que os algoritmos apresentaram desempenhos similares em termos de tempo de processamento, memória consumida e números de padrões detectados. Assim, por possibilitar a escolha do tamanho mínimo da sequência, e por ser mais flexível que os demais algoritmos, o SPAM foi selecionado para análise. Para um limiar de 0,2%, foram encontradas 1.202 sequências frequentes com cinco interações; 258 com seis interações; e 15 com sete interações, resultando em um total de 1.475 padrões com pelo menos cinco interações. As Tabelas 4 e 5 detalham, respectivamente, as cinco sequências mais frequentes e as cinco maiores sequências identificadas.

Sequência	Frequência
4_1, 5_1, 13_1, 14_1, 15_1	72
4_1, 6_1, 8_1, 14_1, 15_1	72
4_1, 5_1, 14_1, 15_1, 15_2	71
4_1, 5_1, 11_1, 11_2, 11_3	71
4_1, 6_1, 14_1, 15_1, 15_2	71

Tabela 5. Cinco sequências mais frequentes

Sequência	Frequência
4_1, 6_1, 6_2, 6_3, 6_4, 6_5, 6_6	52
4_1, 6_1, 6_2, 6_3, 6_4, 6_5, 10_1	52
4_1, 6_1, 6_2, 6_3, 6_4, 8_1, 10_1	50
4_1, 6_1, 6_2, 6_3, 6_4, 14_1, 15_1	50
4_1, 6_1, 6_2, 6_3, 6_4, 6_5, 13_1	49

Tabela 6. Cinco maiores sequências.

As sequências obtidas por meio do algoritmo SPAM mostram resultados similares à estratégia da poda em grafo, evidenciando a dificuldade dos alunos com a questão 6. Também mostram padrões de navegação incomuns pelos problemas, como a transição da questão 5 do tópico de variáveis e atribuição para as questões 13, 14 e 15 de comandos condicionais, presente no sequência mais frequente.

5. Conclusão

O presente trabalho discute como a mineração de sequências pode ser utilizada para extrair padrões de aprendizagem frequentes em um ambiente voltado ao ensino de algoritmos. O conjunto de dados analisado no presente trabalho diz respeito a um total de 9.562 submissões de código, escritas na linguagem de programação Python, realizadas por 229 alunos matriculados em 6 turmas diferentes, para um total de 22 problemas.

A identificação dos padrões sequenciais se deu por meio de duas abordagens distintas. A primeira consistiu na poda do grafo de transições entre questões e tentativas. A segunda compreendeu a utilização de uma biblioteca, denominada SPMF. Após vários testes, as sequências obtidas por meio do algoritmo SPAM, selecionado por permitir a escolha do tamanho da sequência, mostraram resultados similares à estratégia da poda no grafo. Os dados apontam que as questões 6, 7, 11 e 12 são problemáticas e requerem uma atenção especial por parte do professor. Também é importante se estar atento às variações identificadas nos padrões de navegação, pois algumas questões do tópico de variáveis e atribuição não parecem contribuir para o tópico seguinte.

Nesse sentido, os padrões extraídos podem contribuir para a melhoria do ambiente de ensino nos seguintes aspectos: 1. identificação das questões que precisam passar por um processo de revisão, de modo que os alunos consigam superá-las sem que sejam necessárias tantas tentativas; 2. definição da ordem em que as questões devem ser exibidas na interface gráfica, para que os alunos sintam o aumento gradual da dificuldade das questões, mas mesmo assim se sintam instigados a continuar usando a plataforma; 3. compreensão do processo de aprendizagem pelos professores, a partir da visualização de como os alunos interagem com a plataforma, possibilitando a rápida intervenção no sentido de evitar a desmotivação do aluno.

Como trabalhos futuros, vislumbra-se a possibilidade de se integrar a mineração de padrões sequenciais ao ambiente de ensino ora analisado, buscando automatizar o processo de recomendação de problemas com base na navegação dos usuários com comportamentos de aprendizagem similares e, também, como forma de fornecer *feedback* imediato ao professor, conforme o uso do ambiente pelos alunos.

Referências

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE.

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435.

Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.-W., Tseng, V. S., et al. (2014). Spmf: a java open-source pattern mining library. *J. Mach. Learn. Res.*, 15(1):3389–3393.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. (2000). Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. IEEE.

Kang, J., Liu, M., and Qu, W. (2017). Using gameplay data to examine learning behavior patterns in a serious game. *Computers in Human Behavior*, 72:757–770.

Liu, T. and Israel, M. (2022). Uncovering students' problem-solving processes in game-based learning environments. *Computers & Education*, 182:104462.

Lo, D., Khoo, S.-C., and Liu, C. (2008). Efficient mining of recurrent rules from a sequence database. In *Database Systems for Advanced Applications: 13th International Conference, DASFAA 2008, New Delhi, India, March 19-21, 2008. Proceedings 13*, pages 67–83. Springer.

Mirzaei, M. and Sahebi, S. (2019). Modeling students' behavior using sequential patterns to predict their performance. In *Artificial Intelligence in Education: 20th International Conference, AIED 2019, Chicago, IL, USA, June 25-29, 2019, Proceedings, Part II 20*, pages 350–353. Springer.

Molenaar, I. and Wise, A. F. (2022). Temporal aspects of learning analytics–grounding analyses in concepts of time. *The handbook of learning analytics*, pages 66–76.

Slim, A., Heileman, G. L., Al-Doroubi, W., and Abdallah, C. T. (2016). The impact of course enrollment sequences on student success. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 59–65. IEEE.

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology—EDBT'96: 5th International Conference on Extending Database Technology Avignon, France, March 25–29, 1996 Proceedings 5*, pages 1–17. Springer.

Yang, Z., Wang, Y., and Kitsuregawa, M. (2007). Lapin: effective sequential pattern mining algorithms by last position induction for dense databases. In *Advances in Databases: Concepts, Systems and Applications: 12th International Conference on Database*

Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007. Proceedings 12, pages 1020–1023. Springer.

Zaki, M. J. (2000). Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429.

Zheng, J., Li, S., and Lajoie, S. P. (2022). Diagnosing virtual patients in a technology-rich learning environment: A sequential mining of students' efficiency and behavioral patterns. *Education and Information Technologies*, pages 1–17.

Zhou, M., Xu, Y., Nesbit, J. C., and Winne, P. H. (2010). Sequential pattern analysis of learning logs: Methodology and applications. *Handbook of educational data mining*, 107:107–121.