Aplicação de Metaheurísticas Evolutivas na Mineração de Padrões Sequenciais de Aprendizagem em Ambientes de Ensino de Algoritmos

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

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

djefferson.maranhao@gmail.com, carlos.salles@ufma.br

Abstract. *The present work qualitatively analyzes the use of evolutionary metaheuristics in mining learning patterns in a virtual environment designed for teaching algorithms. The central question is how to provide a better learning experience to students on this platform by mining navigation patterns through the content available in this environment. The proposed approach enables educators to comprehensively understand students' learning experiences, providing valuable insights into their navigation patterns and level of student engagement. With this information, educators can promptly identify signs of demotivation and implement specific interventions to keep students interested and engaged on the platform. This study's main contribution demonstrates that this approach can yield valuable insights, such as the need to review a specific problem or provide more practical examples to students.*

Resumo. *O presente trabalho analisa qualitativamente o emprego de metaheurísticas evolutivas na mineração de padrões sequenciais de aprendizagem em um ambiente de ensino de algoritmos. A questão central é como proporcionar uma melhor experiência de aprendizagem aos alunos dessa plataforma mediante a mineração dos padrões de navegação pelo conteúdo disponibilizado nesse ambiente. A abordagem proposta permite ao professor compreender de forma mais ampla a experiência de aprendizagem e o grau de engajamento do aluno. Com esses dados, o professor pode antecipar sinais de desmotivação e intervir para manter o aluno interessado e ativo na plataforma. A principal contribuição deste trabalho é demonstrar que essa abordagem pode gerar insights valiosos, como a necessidade de revisar um problema específico ou fornecer mais exemplos práticos aos alunos.*

1. Introdução

Nos últimos anos, o ensino de programação vem passando por uma transformação significativa, movendo-se do formato tradicional em sala de aula para ambientes de ensino *online*. Este aumento pode ser atribuído, em grande parte, à mudança global para o ensino à distância provocada pela pandemia da COVID-19, que acelerou a adoção de ferramentas e recursos de aprendizagem *online*. Além disso, plataformas especializadas em entrevistas técnicas, como HackerRank e LeetCode, têm promovido a importância da resolução de problemas como um meio para alcançar melhores posições no mercado de trabalho.

DOI: 10.5753/sbie.2024.242581

A acessibilidade destas plataformas democratizou a educação, permitindo que alunos de todo o mundo adquiram competências de programação de qualquer lugar ligado à Internet. Com o aumento do uso de plataformas *online*, o volume de dados gerados por essas ferramentas cresceu exponencialmente. Isso inclui interações dos usuários, progresso no aprendizado, resultados de submissões, entre outros. A análise cuidadosa desse grande volume de dados é essencial para melhorar a experiência de ensino e personalizar o aprendizado de acordo com as necessidades individuais dos estudantes.

Uma técnica promissora para esse tipo de análise é a Mineração de Padrões Sequenciais (MPS), que permite identificar padrões frequentes de comportamento dos alunos ao longo do tempo. Essa técnica pode ser utilizada para prever o desempenho dos estudantes, detectar dificuldades no aprendizado e sugerir intervenções personalizadas. Esses padrões de comportamento frequentes revelam como os alunos interajem com os recursos didáticos, contribuindo assim para a melhoria contínua do ensino online [Zhou et al. 2010, Zhang and Paquette 2023]

Ocorre que os algoritmos de MPS clássicos, como PrefixSpan [Pei et al. 2004], GSP [Srikant and Agrawal 1996] e SPADE [Zaki 2000], utilizam métodos determinísticos para explorar sequências frequentes, aplicando técnicas de poda para reduzir padrões candidatos e aumentar a eficiência. Estes algoritmos garantem exaustividade e precisão na descoberta de padrões, mas enfrentam desafios de escalabilidade em bases de dados muito grandes, onde o número de possíveis sequências cresce exponencialmente.

Nesse cenário, as metaheurísticas mostram-se particularmente relevantes, pois oferecem uma maneira robusta de se explorar grandes espaços de busca. Inspiradas em processos naturais e biológicos, metaheurísticas como Algoritmos Genéticos (AG), Otimização por Colônia de Formigas (OCF), Otimização por Enxame de Partículas (OEP) e Algoritmo de Busca de Harmonia (ABH) têm demonstrado grande potencial na descoberta de padrões em dados sequenciais devido à sua capacidade de escapar de ótimos locais e encontrar soluções quase ótimas em um tempo computacional razoável.

Sendo assim, o objetivo deste trabalho é explorar de forma qualitativa a aplicação de diferentes metaheurísticas evolutivas para a descoberta de padrões sequenciais em um ambiente de ensino de algoritmos. Nesse contexto, discute-se a importância desses padrões para a melhoria das estratégias de ensino e a personalização do aprendizado, apresentando-se um estudo de caso no qual essas técnicas são aplicadas a um conjunto de dados real de interações de alunos em um curso de algoritmos.

O presente artigo está organizado como segue: a Seção 2 apresenta a fundamentação teórica acerca da mineração de padrões sequenciais e das metaheurísticas investigadas; a Seção 3 apresenta os trabalhos que se relacionam a esta pesquisa; a Seção 4 descreve o processo de análise dos dados coletados; e a Seção 5 apresenta a conclusão do trabalho.

2. Mineração de Padrões Sequenciais (MPS)

A mineração de padrões sequenciais (MPS) é uma tarefa essencial na descoberta de conhecimento em grandes bases de dados. Seu objetivo principal é identificar subsequências frequentes em um conjunto de sequências. Algoritmos clássicos, como o PrefixSpan [Pei et al. 2004], GSP (*Generalized Sequential Patterns*) [Srikant and Agrawal 1996] e

SPADE (*Sequential PAttern Discovery using Equivalent Class*) [Zaki 2000], utilizam abordagens determinísticas para explorar o espaço de busca de sequências frequentes.

Esses métodos frequentemente empregam técnicas de poda para reduzir o número de padrões candidatos a serem avaliados, aumentando assim a eficiência da mineração. Por exemplo, o PrefixSpan evita a geração explícita de padrões candidatos ao projetar o banco de dados de entrada em sub-bancos de dados mais gerenciáveis, enquanto o GSP aplica uma estratégia de junção e poda baseada no suporte mínimo [Pei et al. 2004, Agrawal and Srikant 1995].

Embora esses algoritmos sejam bem estabelecidos e amplamente utilizados devido à sua capacidade de garantir exaustividade e precisão na descoberta de padrões sequenciais, eles enfrentam desafios de escalabilidade quando aplicados a bases de dados muito grandes ou complexas, onde o número de possíveis sequências cresce exponencialmente [Zaki 2001].

Em contraste, a utilização de metaheurísticas na mineração de padrões sequenciais surge como uma abordagem promissora para lidar com a complexidade e escalabilidade das bases de dados modernas. Metaheurísticas como algoritmos genéticos, algoritmos de colônia de formigas e busca por harmonia são inspiradas em processos naturais e biológicos e têm sido aplicadas com sucesso para resolver problemas complexos de otimização.

Por exemplo, algoritmos genéticos utilizam operadores de seleção, cruzamento e mutação para explorar o espaço de busca e evoluir uma população de soluções candidatas. De forma semelhante, algoritmos de colônia de formigas utilizam a metáfora de colônias de formigas para buscar caminhos otimizados, baseando-se em feromônios virtuais que guiam a exploração [Karimi-Mamaghan et al. 2021].

2.1. Metaheurísticas para Mineração de Padrões Sequenciais

As metaheurísticas são estratégias de otimização de propósito geral especialmente úteis em situações em que métodos exatos de busca são inviáveis devido à alta dimensionalidade ou à natureza complexa do espaço de busca. Elas não garantem encontrar a solução ótima, mas podem encontrar boas soluções em um tempo razoável. Outra vantagem das metaheurísticas é sua flexibilidade e adaptabilidade, permitindo ajustes finos e hibridizações para diferentes tipos de problemas [Parpinelli and Lopes 2011].

2.1.1. Algoritmo Genético

O Algoritmo Genético (AG) foi proposto por J. H. Holland em 1975, inspirado na Teoria da Evolução de Darwin [Pires and Silva 2016]. Os conceitos evolutivos são usados para definir uma técnica de busca para resolver problemas de otimização. Nessa técnica, uma população de soluções candidatas (inicializada aleatoriamente) evolui por meio da seleção dos indivíduos mais aptos (melhores soluções candidatas), *crossover* e mutação.

O processo começa com um conjunto de indivíduos (população), onde cada indivíduo representa uma solução candidata para o problema. Durante a seleção dos pais, um conjunto de indivíduos é escolhido para reprodução. Os métodos mais comuns de seleção incluem roleta, classificação e torneio.

Os indivíduos selecionados são recombinados por meio de *crossover* para gerar novos indivíduos. Uma das abordagens mais simples é o cruzamento de pontos, onde um ou mais pontos de cruzamento são escolhidos aleatoriamente e o filho é criado pela troca dos genes entre os pais.

A mutação envolve pequenas modificações nas soluções candidatas geradas. Nesta etapa, um ou mais bits são alterados de acordo com uma probabilidade de mutação. A mutação é fundamental para manter a diversidade genética da população e evitar a convergência prematura para soluções subótimas.

Por fim, as novas soluções candidatas geradas pelo *crossover* e pela mutação são então avaliadas em relação à função objetivo, e a população atual é atualizada. Muitas vezes, os melhores indivíduos da população são mantidos, e até 10% das piores soluções candidatas podem ser preservadas para manter a diversidade. O algoritmo termina quando o critério de parada é alcançado.

2.1.2. Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas (OEP), proposta por Russell Eberhart e James Kennedy, é inspirada nos comportamentos coletivos observados em bandos de pássaros e cardumes de peixe [Eberhart and Kennedy 1995]. Este algoritmo utiliza informações tanto locais quanto globais para guiar o movimento e melhorar a qualidade das soluções candidatas representadas pelas partículas.

Cada partícula no OEP é caracterizada por três componentes principais: velocidade, um vetor de valores contínuos que indica a direção e a magnitude do movimento da partícula; posição, um vetor de valores contínuos que representa a solução candidata atual; e melhor local (P_{best}), a melhor posição encontrada pela partícula até o momento. O algoritmo também armazena o Melhor Global (G_{best}), que é a melhor posição encontrada por qualquer partícula no enxame.

O movimento das partículas é influenciado por suas experiências individuais (P_{best}) e pela experiência coletiva do enxame (G_{best}). A cada iteração, as velocidades das partículas são ajustadas levando em conta suas velocidades anteriores, a diferença entre suas posições atuais e suas melhores posições individuais, além da diferença entre suas posições atuais e a melhor posição global encontrada pelo enxame. Esses ajustes consideram fatores como a inércia da partícula e os coeficientes que determinam a influência das experiências individual e coletiva.

O processo de atualização continua até que um critério de parada seja atingido, que pode ser um número máximo de iterações ou a convergência para uma solução aceitável.

2.1.3. Otimização por Colônia de Formigas

A Otimização por Colônia de Formigas (OCF) foi proposta por Marco Dorigo em 1992, inspirada no comportamento de busca de alimentos das formigas [Dorigo 1992]. As formigas depositam feromônio ao longo de seu caminho, criando trilhas que outras formigas seguem, com a intensidade do feromônio influenciando a probabilidade de escolha das

trilhas.

No OCF, uma população de formigas é utilizada para construir soluções para um problema de otimização. Cada formiga constrói uma solução ao adicionar componentes à solução parcial com base na quantidade de feromônio e em uma medida heurística da qualidade do componente. O algoritmo inicia com a definição dos parâmetros, como o número de formigas, o coeficiente de evaporação do feromônio (ρ), e os parâmetros de controle (α e β). Além disso, a quantidade de feromônio em todas as arestas é inicializada com um valor inicial τ_0.

Enquanto constroem suas soluções, as formigas escolhem os componentes de maneira probabilística, considerando tanto a quantidade de feromônio presente nas arestas (τ_{ij}) quanto a informação heurística associada a elas (η_{ij}). A probabilidade P_{ij}^k de uma formiga k escolher um determinado componente i após outro componente j é calculada com base em uma combinação desses fatores, ajustada pelos parâmetros de controle.

Depois que todas as formigas completam suas soluções, a quantidade de feromônio nas arestas é atualizada para refletir a qualidade das soluções encontradas. Esse processo de atualização leva em conta a evaporação natural do feromônio e a quantidade depositada por cada formiga, reforçando as arestas que fazem parte de soluções melhores. A evaporação é essencial para evitar a convergência prematura, garantindo que a exploração continue ao longo das iterações.

O processo de construção de soluções e atualização de feromônio é repetido até que um critério de parada seja atingido, como um número máximo de iterações ou a convergência para uma solução aceitável.

2.1.4. Algoritmo de Busca de Harmonia

O Algoritmo de Busca de Harmonia (ABH) é uma heurística de busca baseada no processo de improvisação dos músicos de jazz [Geem et al. 2001]. No jazz, os músicos tentam ajustar os tons de seus instrumentos, de forma que as harmonias gerais sejam otimizadas devido a aspectos estéticos. Começando com algumas harmonias, eles tentam alcançar melhores harmonias através da improvisação.

Essa analogia pode ser usada para derivar heurísticas de busca, que podem ser usadas para otimizar uma determinada função objetivo em vez de harmonias. Aqui os músicos são identificados com as variáveis de decisão e as harmonias correspondem às soluções. Tal como os músicos de jazz criam novas harmonias através da improvisação, o algoritmo ABH cria iterativamente novas soluções baseadas em soluções passadas e em modificações aleatórias.

O algoritmo ABH inicializa a *Harmony Memory* (HM) com soluções geradas aleatoriamente. O número de soluções armazenadas na memória HM é definido pelo *Harmony Memory Size* (HMS). Então, iterativamente, uma nova solução é criada como segue. Cada variável de decisão é gerada considerando a memória e uma possível modificação adicional ou por seleção aleatória.

Os parâmetros que são utilizados no processo de geração de uma nova solução são chamados *Harmony Memory Considering Rate* ($HMCR$) e *Pitch Adjusting Rate* (PAR).

Cada variável de decisão é definida como o valor da variável correspondente de uma das soluções na HM com probabilidade de HMCR, e uma modificação adicional deste valor é realizada com probabilidade de PAR.

Caso contrário, com probabilidade de $1-HMCR$, a variável de decisão é definida com um valor aleatório. Após a criação de uma nova solução, ela é avaliada e comparada com a pior solução da memória. Se o seu valor objetivo for melhor que o da pior solução, substitui a pior solução no HM. Este processo é repetido até que um critério de parada seja satisfeito.

3. Trabalhos Relacionados

As metaheurísticas AG, OCF, OEP e ABH têm sido aplicadas no contexto educacional para personalizar e otimizar a experiência de aprendizagem dos estudantes. Os estudos apresentados a seguir propõem abordagens promissoras para enfrentar os desafios de sobrecarga cognitiva e desorientação dos alunos, proporcionando um aprendizado mais eficiente e personalizado.

Em [Bhaskar et al. 2010] é descrito um sistema que utiliza AGs para criar esquemas de aprendizado adaptativos baseados no contexto específico de cada aluno. O método considera múltiplos parâmetros, como perfil, preferências e infraestrutura, para gerar conteúdos de ensino personalizados. A eficácia foi demonstrada com um curso de redes de computadores, onde o algoritmo gerou esquemas de aprendizagem adaptados às necessidades e preferências individuais dos alunos.

Em [Sharma et al. 2012], é proposto um algoritmo baseado na metaheurística OCF, que avalia o nível de um aprendiz e recomenda os conceitos apropriados para ele. Esse algoritmo é sensível às mudanças nos comportamentos de aprendizagem de cada aluno e ajusta suas estratégias para recomendar o próximo conceito de acordo com a necessidade. Os comportamentos dos alunos anteriores são capturados e utilizados para recomendar conteúdo a futuros alunos.

Em [Li et al. 2012] é detalhado o uso de metaheurísticas para compor cursos personalizados, atendendo às necessidades individuais dos alunos. O processo utiliza as metaheurísticas AG e OEP na etapa de composição do curso personalizado. Os experimentos realizados mostraram que, com até 300 materiais de aprendizagem, o OEP se destaca pela eficiência em termos de tempo e número de gerações necessárias para convergir para uma solução ótima. Com mais de 300 materiais, o AG foi mais eficiente.

Por fim, em [Hnida et al. 2016], o algoritmo ABH foi sugerido como abordagem ao problema de sequenciamento curricular. Esse algoritmo é inspirado no processo de improvisação musical, em que um grupo de músicos improvisa o tom de seus instrumentos, buscando a harmonia perfeita. A pesquisa adapta o ABH para sequenciar objetos de aprendizagem de maneira a maximizar a relevância dos conteúdos apresentados aos alunos, considerando o nível de conhecimento dos alunos e as inter-relações de conteúdos.

Esses estudos mostram que a aplicação de metaheurísticas pode trazer benefícios significativos para a personalização das experiências de aprendizagem. O presente trabalho expande essas investigações ao explorar o uso dessas metaheurísticas especificamente no contexto do ensino de algoritmos, buscando identificar padrões que possibilitem melhorar as estratégias de ensino e a personalização do aprendizado.

4. Metodologia

Esta seção descreve o processo de análise de dados adotado no presente estudo. A Seção 4.1 detalha o processo de coleta dos dados. A Seção 4.2 discute as atividades realizadas de limpeza e transformação dos dados relacionais em dados sequenciais. A Seção 4.3 explora brevemente o conjunto de dados, proporcionando um entendimento mais claro ao leitor. A Seção 4.4 aborda a técnica de mineração de padrões sequenciais apoiada por metaheurísticas evolutivas. Por fim, a Seção 4.5 discute os resultados obtidos.

Figura 1. Metodologia de análise.

4.1. Obtenção dos Dados

A coleta foi realizada mediante consulta direta à base de dados da plataforma de ensino utilizada na disciplina de Algoritmos I do Curso de Ciência da Computação da Universidade Federal do Maranhão. Os dados foram obtidos a partir do cruzamento entre as tabelas de questões e submissões, resultando em um conjunto de dados composto pelas seguintes colunas: $usuario$, $questao$, $conceito$, $turma$, $tipo\_resultado$, $linguagem$, $codigo$, $data\_criacao$, $tempo\_inicial$, $tempo\_final$ e $resultado$.

A plataforma funciona internamente como um sistema de juiz *online*, sendo capaz de compilar os códigos-fonte submetidos pelos alunos e executá-los contra casos de teste previamente definidos, avaliando o resultado produzido pelas submissões em: sucesso, se nenhum caso de teste falhar; erro de compilação, se houver qualquer falha no processo de compilação; tempo limite excedido, se o código for incapaz de gerar os resultados esperados no tempo previsto; ou erro de execução, se algum caso de teste falhar.

No período de 2021 a 2023, quando os dados foram coletados, a plataforma ofereceu material teórico e atividades práticas de programação acerca de quatro tópicos: variáveis e atribuição; comandos condicionais; laços de repetição; vetores e listas. Durante esse período, os alunos puderam navegar livremente pelos conteúdos disponibilizados na plataforma, sem que lhes fosse oferecido qualquer tipo de suporte adicional à navegação. Além disso, a disciplina foi lecionada sempre pelo mesmo professor.

4.2. Limpeza e Transformação dos Dados

A etapa de limpeza envolveu a remoção dos alunos com quantidades de interações muito discrepantes (*outliers*), identificados por meio do método do intervalo interquartílico. Antes da limpeza, o primeiro quartil (Q_1) da distribuição de interações por aluno era de 21, enquanto o terceiro quartil (Q_3) era de 72, resultando em um $IQR = Q_3 - Q_1 = 72 - 21 = 51$. Foram considerados *outliers* quaisquer observações menores que $Q_1 - 1,5 \times IQR = 21 - 1,5 \times 51 = -55,5$ ou maiores que e $Q_3 + 1,5 \times IQR = 72 + 1,5 \times 51 = 148,5$. Como resultado desse processo, 16 usuários e 4.029 submissões foram excluídos da análise.

Em seguida, os registros de submissão foram convertidos de um formato relacional para uma estrutura de lista. Isso envolveu primeiro agrupar as submissões por usuário

e, em seguida, ordenar cada grupo pela data e hora das interações, criando sequências temporais de atividades. O objetivo dessa transformação é simplificar a identificação de padrões de comportamento e outros fatores relevantes ao desempenho dos alunos, conforme será apresentado nas seções seguintes.

4.3. Análise Exploratória e Visualização dos Dados

Esta etapa tem como principal objetivo entender a estrutura dos dados, buscando identificar padrões e detectar anomalias, assim como resumir as principais características dos dados com a ajuda de métodos visuais. Após a etapa de limpeza, o conjunto de dados possui um total de 13.916 submissões realizadas por 313 alunos de 7 turmas (semestres) diferentes, todas escritas em Python.

Estão presentes no conjunto de dados um total de 68 (sessenta e oito) problemas, dos quais 7 são sobre variáveis e atribuição; 12 sobre comandos condicionais; 29 sobre laços de repetição; e 20 sobre vetores e listas, conforme apresentado na Tabela 1. As questões possuem níveis de dificuldade variados, partindo de um simples "Hello World" até o cálculo de números primos em um determinado intervalo.

Tabela 1. Distribuição das questões por tópico.

Tópico	Identificadores
Variáveis e Atribuição	3, 4, 6, 7, 8, 12, 80, 82, 84, 88, 206, 210
Comandos Condicionais	5, 10, 11, 13, 14, 15, 18, 19, 30, 94, 106, 216
Laços de Repetição	16, 17, 26, 31, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 236, 238, 240, 242, 244, 246, 248, 250
Vetores e Listas	25, 27, 29, 56, 57, 58, 59, 60, 65, 67, 69, 71, 72, 73, 74, 75, 76, 254, 256, 262

No conjunto de dados, podem ser encontradas 6.557 soluções bem-sucedidas; 5.155 submissões com erros de compilação; 2.140 resultaram em erros de tempo de execução; e 64 submissões que excederam o limite permitido. Os tipos de resultado estão distribuídos pelas questões conforme representado na Figura 2.

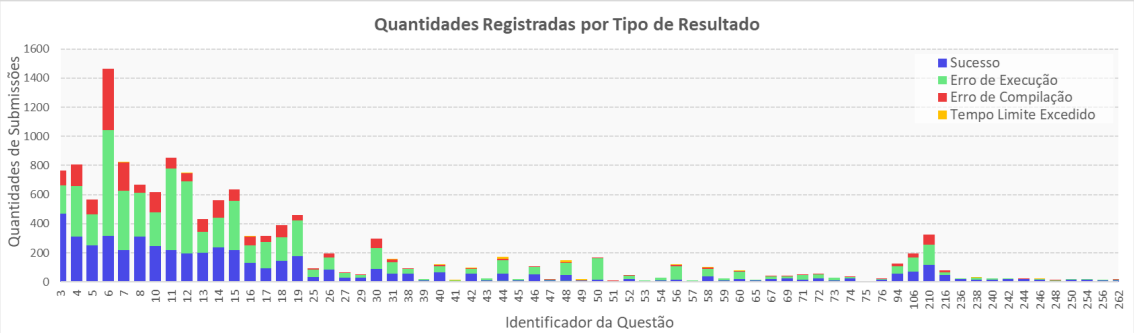

Figura 2. Distribuição dos tipos de resultado por questão.

Em média, cada aluno teve 44,46 interações, mas com uma grande variação, refletida no desvio padrão de 32,95. O aluno menos participativo teve apenas uma interação, enquanto o mais ativo teve 184. Em relação às questões, cada uma recebeu em média 400,69 interações, com um desvio padrão ainda maior, de 404,27. A questão menos popular teve 16 submissões, enquanto a mais popular teve 1.801. Esses dados demonstram uma grande diversidade tanto na participação dos alunos quanto no interesse pelas questões.

4.4. Modelagem do Problema

A abordagem adotada neste trabalho baseia-se na construção de um grafo, onde os nós representam as questões com as quais os alunos da plataforma de ensino interagiram. Cada nó é enriquecido com informações sobre o número de tentativas realizadas para resolver aquela questão específica. As arestas, por sua vez, indicam o número de transições de uma questão para outra, criando assim uma representação gráfica das trajetórias de aprendizado.

Para extrair padrões sequenciais desse grafo, foram geradas soluções candidatas utilizando quatro metaheurísticas distintas: AG, OEP, OCF e ABH. A função objetivo utilizada para avaliar a qualidade dessas soluções foi a maximização do somatório das arestas do grafo. Essa função objetivo, quando aplicada a uma solução candidata, gera um valor chamado *fitness*, que reflete quão consistente e frequente é a trajetória de aprendizado identificada entre os alunos.

Cada metaheurística foi configurada para buscar sequências de tamanho 7, ao longo de 10.000 gerações. As Figuras 3, 4 e 5 detalham, respectivamente, o valor da função de *fitness* por geração, o tempo de execução (em segundos) e a memória consumida (em MB).

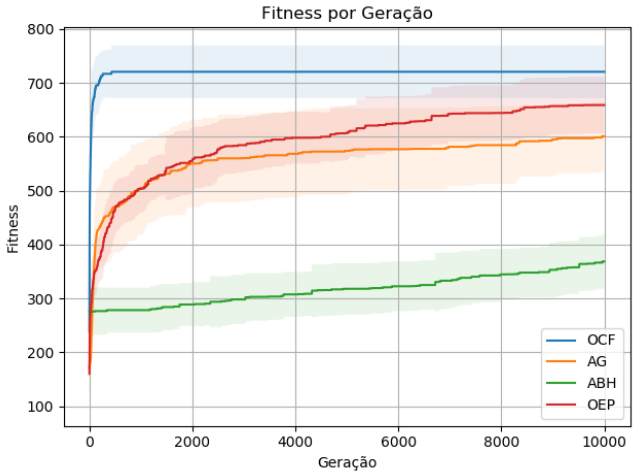

Figura 3. *Fitness* vs. Geração

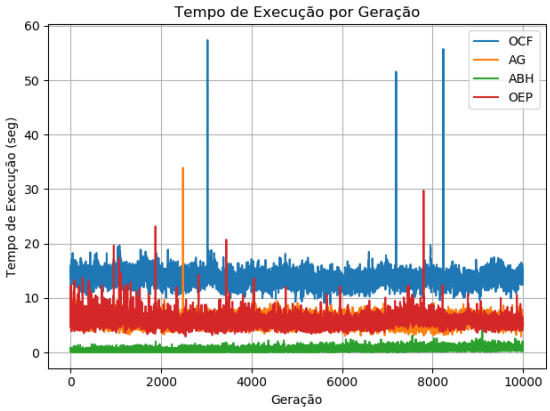

Figura 4. Tempo de Execução vs. Geração

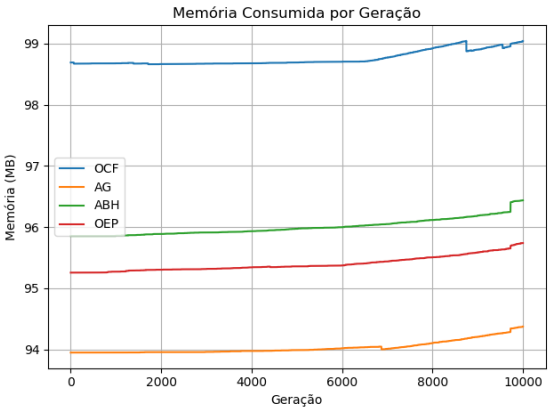

Figura 5. Memória Consumida vs. Geração

Cada algoritmo foi executado 30 vezes no *dataset* para garantir a robustez dos resultados. Os experimentos foram realizados em um ambiente com as seguintes especificações: CPU i7-8550U @ 1.80GHz 1.99 GHz, 16 GB de RAM DDR4, SSD de 512 GB, rodando Windows 11 x64. O resultado dos experimentos é um banco de sequências ordenado com base no somatório dos pesos das arestas, conforme representado na Fig. 6:

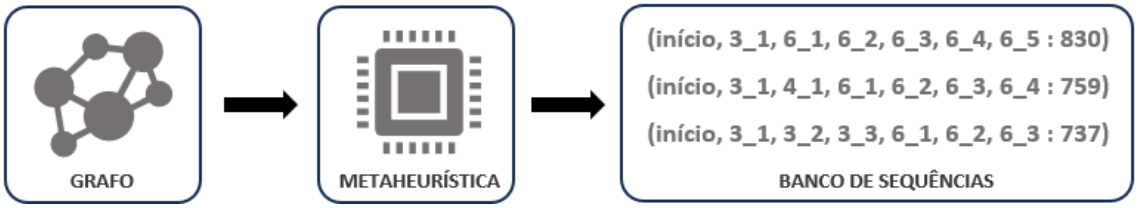

Figura 6. Processo de execução das experimentos.

4.5. Interpretação dos Resultados

Os resultados obtidos na etapa anterior indicam que o algoritmo OCF apresenta um desempenho superior em termos de *fitness* em comparação com os demais, para o conjunto de dados em análise. Na prática, isso significa que o algoritmo é capaz de descobrir a trajetória de aprendizagem mais adotada pelos alunos de forma mais rápida. Outros algoritmos que também mostraram bom desempenho foram o OEP e o AG, que conseguiram encontrar sequências relevantes em um número adequado de gerações. Em contrapartida, o algoritmo ABH apresentou um desempenho consideravelmente inferior, necessitando cerca de 50.000 gerações para alcançar a convergência.

Em termos de tempo de execução e consumo de memória, o algoritmo OCF apresenta desvantagens em relação aos concorrentes. Seu tempo de execução é quase duas vezes maior que o dos outros algoritmos, e seu consumo de memória também é mais elevado. Por outro lado, os algoritmos OEP e AG conseguem alcançar resultados similares em termos de *fitness*, mas com menor uso de recursos. O algoritmo ABH destaca-se pelo menor tempo de processamento, pois realiza apenas um ajuste na população a cada geração, substituindo a pior harmonia por uma solução candidata melhor, quando esta é encontrada.

A Tabela 2 detalha os padrões mais frequentes identificados pelos algoritmos. Os resultados mostram que os alunos enfrentam grandes dificuldades para superar as questões 6, 7, 11 e 12. Essas questões se destacam em relação às demais, justificando a dificuldade adicional enfrentada pelos alunos. Por exemplo, nas questões 6 e 7, conforme mostra a Tabela 3, o professor exige a formatação da saída com um número específico de casas decimais, porém, a plataforma não fornece exemplos de como fazer isso. As questões 11 e 12 envolvem não apenas comandos condicionais, mas também conceitos matemáticos de sistemas lineares, acrescentando um nível extra de complexidade.

Tabela 2. Dez maiores sequências.

Id.	Padrão Sequencial	Soma	Id.	Padrão Sequencial	Soma
1	$inicio, 3\_1, 6\_1, 6\_2, 6\_3, 6\_4, 6\_5$	830	6	$6\_1, 6\_2, 6\_3, 6\_4, 6\_5, 6\_6, 6\_7$	717
2	$inicio, 3\_1, 4\_1, 6\_1, 6\_2, 6\_3, 6\_4$	759	7	$inicio, 3\_1, 6\_2, 6\_3, 6\_4, 6\_5, 6\_6$	699
3	$inicio, 3\_1, 3\_2, 3\_3, 6\_1, 6\_2, 6\_3$	737	8	$5\_1, 6\_1, 6\_2, 6\_3, 6\_4, 6\_5, 6\_6$	683
4	$inicio, 3\_1, 6\_1, 6\_2, 6\_3, 6\_4, 7\_1$	728	9	$inicio, 3\_1, 4\_1, 12\_1, 12\_2, 12\_3, 12\_4$	663

5	*inicio*, 3_1, 6_1, 6_2, 6_3, 6_4, 7_2	723	10	*inicio*, 3_1, 4_1, 5_1, 11_1, 11_2, 11_3	620

Tabela 3. Enunciados das Questões

Id	Enunciado	Tópico
3	Imprima a mensagem "Hello World!" na tela. Atenção para caracteres maiúsculos e minúsculos e para a exclamação, que também precisa ser impressa.	Variáveis e Atribuição
4	Escreva um programa que pede para o usuário digitar um inteiro. O programa deve exibir como resultado o dobro desse número inteiro.	Variáveis e Atribuição
5	Escreva um programa que pede para o usuário informar a medida da base e da altura de um retângulo (ambos inteiros). Depois, o programa imprime em uma linha isolada o valor da área desse retângulo.	Comandos Condicionais
6	Faça um programa que pede para o usuário informar a medida da base e da altura de um retângulo (ambos reais, com uma casa decimal). Depois, o programa imprime em uma linha isolada o valor da área desse retângulo, com uma casa decimal de precisão.	Variáveis e Atribuição
7	Crie um programa que receba dois números inteiros do usuário: o primeiro é o número de gols feitos por uma determinada equipe no campeonato e o segundo é o número de gols sofridos. Visto que o saldo de gols é a diferença entre o número de gols feitos e gols sofridos, e que o campeonato tem 38 jogos, calcule a média do saldo de gols de cada equipe por jogo.	Variáveis e Atribuição
11	Faça um programa em que o usuário informa dois inteiros, que chamaremos de SOMA e SUBTRAÇÃO. A SOMA é a soma de maçãs e bananas que o usuário comeu durante a semana. Já SUBTRAÇÃO corresponde à diferença entre a quantidade de bananas e de maçãs que aquele usuário comeu durante a semana. Dados os dois inteiros SOMA e SUBTRAÇÃO, seu programa deve exibir a quantidade de maçãs que o usuário comeu durante a semana.	Comandos Condicionais
12	Faça um programa em que o usuário informa três inteiros em três linhas isoladas. O primeiro inteiro corresponde à soma de maçãs e laranjas que ele comeu durante a semana. Já o segundo inteiro corresponde à soma de maçãs e bananas que ele comeu. Por fim, o terceiro inteiro corresponde à soma de laranjas e bananas que ele comeu. Dadas as três somas, imprima em três linhas isoladamente a quantidade de maçãs (primeira linha), laranjas (segunda linha) e bananas (terceira linha) que o usuário comeu durante aquela semana.	Variáveis e Atribuição

Conforme ilustrado na Figura 7, os alunos geralmente resolvem a questão 3, "Hello World", e em seguida avançam diretamente para a questão 6, "Área de um Retângulo Real", que exige a formatação da saída com duas casas decimais. Outra característica observada é que os alunos tentam responder às questões do tópico "Comandos Condicionais" (questões 5 e 11) sem terem concluído as questões de "Variáveis e Atribuição", o que não condiz com o comportamento desejado pelo professor. A detecção desse tipo de padrão é essencial para que o professor possa intervir e evitar um eventual processo de desmotivação causado pelos sucessivos erros em uma mesma questão. Essa intervenção pode incluir a revisão das questões, o fornecimento de exemplos práticos ou até mesmo a reorganização das questões.

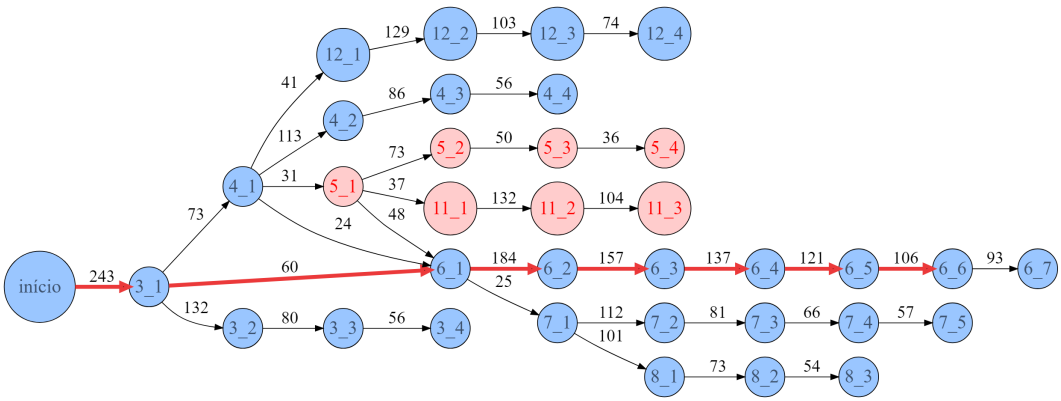

Figura 7. Grafo obtido a partir das maiores sequências.

5. Conclusão

O presente trabalho investiga como metaheurísticas evolutivas podem ser utilizadas para realizar a mineração de padrões sequenciais em ambientes voltados ao ensino de algoritmos, extraindo assim as trajetórias de aprendizagem mais frequentes. O conjunto de dados analisado no presente trabalho diz respeito a um total de 13.916 submissões de código, escritas na linguagem de programação Python, realizadas por 313 alunos matriculados em 7 turmas diferentes, para um total de 68 problemas.

A abordagem deste trabalho baseia-se na construção de um grafo para modelar a interação dos alunos com questões de uma plataforma de ensino. Os nós representam as questões, enriquecidos com informações sobre o número de tentativas, e as arestas indicam as transições entre questões, criando trajetórias de aprendizado. Quatro metaheurísticas (AG, OEP, OCF, ABH) foram usadas para gerar soluções candidatas, buscando maximizar o somatório das arestas do grafo. As soluções foram avaliadas ao longo de 10.000 gerações, detalhando o valor de *fitness* por geração, tempo de execução e memória consumida. Experimentos foram realizados 30 vezes, resultando em um banco de sequências ordenado pelos pesos das arestas.

Os resultados indicam que o algoritmo OCF teve o melhor desempenho em identificar trajetórias de aprendizado, embora com maior tempo de execução e consumo de memória. O OEP e AG também mostraram bom desempenho, com menor uso de recursos, enquanto o ABH necessitou de mais gerações para convergir. Os padrões mais frequentes revelaram dificuldades dos alunos nas questões 6, 7, 11 e 12, muitas vezes devido à complexidade adicional não exemplificada pela plataforma.

Os padrões extraídos podem melhorar o ambiente de ensino de várias maneiras. Primeiramente, ajudam a identificar quais questões precisam de revisão, facilitando para que os alunos as superem com menos tentativas. Além disso, permitem definir uma sequência ideal para a apresentação das questões na interface gráfica, garantindo que os alunos percebam um aumento gradual na dificuldade e se mantenham motivados a continuar usando a plataforma. Finalmente, oferecem ao professor uma compreensão mais profunda do processo de aprendizagem, mostrando como os alunos interagem com a plataforma e permitindo intervenções rápidas para evitar a desmotivação.

No entanto, o trabalho apresenta algumas limitações que devem ser consideradas. Primeiramente, o tempo de execução elevado do algoritmo OCF pode ser um obstáculo em aplicações em larga escala, onde a rapidez na obtenção de resultados é crucial. Além disso, a análise foi limitada a um conjunto específico de dados e questões, o que pode restringir a generalização dos resultados para outros contextos educacionais ou diferentes plataformas de ensino. Também, o consumo de memória elevado pode limitar a aplicabilidade em sistemas com recursos computacionais mais restritos.

Como trabalhos futuros, vislumbra-se a integração da mineração de padrões sequenciais utilizando metaheurísticas evolutivas ao ambiente de ensino analisado, com o objetivo de automatizar a recomendação de problemas com base na navegação de usuários com comportamentos de aprendizagem similares. Além disso, essa integração pode fornecer *feedback* imediato ao professor, permitindo ajustes conforme os alunos utilizam a plataforma.

Referências

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

Bhaskar, M., Das, M. M., Chithralekha, T., and Sivasatya, S. (2010). Genetic algorithm based adaptive learning scheme generation for context aware e-learning. *International Journal on Computer Science and Engineering*, 2(4):1271–1279.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. Phd thesis, Politecnico di Milano, Milan, Italy.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Nagoya, Japan. IEEE.

Geem, Z., Kim, J., and et al. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60.

Hnida, M., Idrissi, M. K., and Bennani, S. (2016). Adaptive teaching learning sequence based on instructional design and evolutionary computation. In *2016 15th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–6. IEEE.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A., and Talbi, E.-G. (2021). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422.

Li, J.-W., Chang, Y.-C., Chu, C.-P., and Tsai, C.-C. (2012). A self-adjusting e-course generation process for personalized learning. *Expert Systems with Applications*, 39(3):3223–3232.

Parpinelli, R. S. and Lopes, H. S. (2011). New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 215–224. IEEE.

Pires, P. and Silva, J. (2016). *Intelligent Systems: Concepts and Applications*. Springer, Berlin.

Sharma, R., Banati, H., and Bedi, P. (2012). Adaptive content sequencing for e-learning courses using ant colony optimization. In *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011*, pages 579–590. Springer.

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.

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.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):31–60.

Zhang, Y. and Paquette, L. (2023). Sequential pattern mining in educational data: The application context, potential, strengths, and limitations. In *Educational Data Science: Essentials, Approaches, and Tendencies: Proactive Education based on Empirical Big Data Evidence*, pages 219–254. Springer.

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.