Sistema de Representação 3D de Perfis para Análise Temporal da Aprendizagem de Programação e Composição de Rubricas

Ádler Oliveira Silva Neves [1] **Márcia Gonçalves de Oliveira** [2], **Mônica Ferreira Silva Lopes** [1]

[1] Instituto Federal do Espírito Santo (Ifes) - Campus Serra

[2]Centro de Referência em Formação e Educação a Distância (Cefor)
Instituto Federal do Espírito Santo (Ifes) - Vitória - ES

clickmarcia@gmail.com

Abstract. *This work presents an online system of 3D representation of learning profiles that maps, for each student, the programming codes he developed along a course in selected variables from a broad set of software metrics that quantify effort and quality of programming. Applying this profile representation, the proposed system offers the following functionalities: generation of student time lines to check the evolution of evaluation variables in a sequence of course workout solutions, different visualizations of these variables, and selection of sample codes representative sources for composition of rubrics. The proposed system is, therefore, an important tool to help teachers of programming in the decision making of evaluation and in the monitoring of the learning of their students.*

Resumo. *Este trabalho apresenta um sistema online de representação 3D de perfis de aprendizagem que mapeia, para cada estudante, os códigos de programação por ele desenvolvidos ao longo de um curso em variáveis selecionadas a partir de um amplo conjunto de métricas de software que quantificam esforço e qualidade de programação. Aplicando essa representação de perfis, o sistema proposto oferece as seguintes funcionalidades: geração de linhas do tempo dos estudantes para análise da evolução das variáveis de avaliação em uma sequência de soluções de exercícios de um curso, diferentes visualizações dessas variáveis e seleção de exemplos de códigos-fontes representativos para composição de rubricas. O sistema proposto apresenta-se, portanto, como uma importante ferramenta para auxiliar professores de programação na tomada de decisões de avaliação e no acompanhamento da aprendizagem de seus alunos.*

1. Introdução

A análise da aprendizagem de programação com as finalidades de assistir e qualificar um processo de aprendizagem do seu início ao fim representa uma onerosa tarefa para professores de programação, uma vez que a prática de programação assistida demanda muito tempo e esforço na correção de atividades, principalmente quando estas são aplicadas em grande quantidade e em turmas numerosas. Dessa forma, realizar durante um curso uma análise de aprendizagem que possibilite comparar soluções de programação desenvolvidas por diferentes estudantes e verificar como as soluções de um estudante evoluem ao longo do tempo representam um verdadeiro desafio para a avaliação de programação.

DOI: 10.5753/cbie.sbie.2018.1573

Embora já existam várias soluções para representar e comparar perfis de estudantes de programação [De Oliveira et al. 2013, Novais et al. 2016], há poucas soluções tecnológicas para uma análise temporal da aprendizagem desses estudantes durante um curso de programação.

Uma proposta mais recente de análise da aprendizagem de programação consiste em mapear códigos-fontes em métricas de software que quantificam esforço e qualidade de programação [Neves et al. 2017]. Através dessas métricas, para cada atividade de programação, é possível comparar soluções de estudantes sob diferentes variáveis para identificar classes de soluções, dificuldades de aprendizagem em comum, boas práticas de programação e até plágios.

Embora a proposta de [Neves et al. 2017] possibilite comparar perfis de estudantes de uma turma em cada atividade de programação, é trabalhoso para um professor por meio desse instrumento verificar como essas métricas de avaliação evoluem ao longo do tempo, isto é, a cada atividade de um curso, para cada estudante. Realizar esse tipo de acompanhamento permite ao professor de programação identificar em que pontos do curso os alunos se desenvolvem melhor em seus processos de aprendizagem e onde começam a apresentar as dificuldades de aprendizagem.

Com o objetivo de suprir essa necessidade oferecendo a professores de programação um instrumento para acompanhar o processo de aprendizagem de seus alunos, este trabalho estende a proposta de [Neves et al. 2017] gerando visualizações 3D de perfis de estudantes mapeados em métricas de software selecionadas. Essas métricas de avaliação caracterizam eficiência, estilo e esforço de programação de cada estudante a cada solução de programação por ele desenvolvida ao longo de um curso.

Além da representação 3D para análise de aprendizagem, esse sistema seleciona dinamicamente amostras de soluções de programação para um professor pontuar até encontrar um conjunto representativo de rubricas para informar critérios de avaliação. Essa funcionalidade poderá contribuir posteriormente para seleção de treino representativo de sistemas de avaliação automática de exercícios de programação.

A principal contribuição deste trabalho para a Informática na Educação é propor um instrumento de apoio à tomada de decisões de avaliação no domínio da programação, possibilitando aos professores a análise e monitoramento da aprendizagem de seus alunos a cada atividade de programação sob um amplo leque de variáveis, antecipando-se às possibilidades de fracassos de desempenhos.

Para apresentar os fundamentos e funcionalidades do sistema proposto, este trabalho está organizado conforme a ordem a seguir. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a arquitetura do sistema com as representações 3D de perfis e a seleção de representações de rubricas. Na Seção 4, destacamos alguns resultados. Na Seção 5, concluimos este trabalho destacando os principais resultados, os trabalhos futuros e as considerações finais.

2. Trabalhos relacionados

Os principais trabalhos relacionados a nossa proposta, além do trabalho de [Neves et al. 2017] que apresentamos no início, são os instrumentos de visualização de perfis de estudantes de programação de [Oliveira et al. 2017], a estratégia de reconheci-

mento de perfis por métricas de análise de codigos-fontes de [Novais et al. 2016], o modelo de seleção de características de [Spalenza et al. 2016], o reconhecimento automático de representações de rubricas de [Gonçalves de Oliveira et al. 2018], o sistema de reconhecimento de rubricas com redução de dimensionalidde de [Olmos et al. 2016] e o estudo de [Tang et al.] envolvendo descoberta de padrões longitudinais.

O trabalho de [Oliveira et al. 2017] apresenta alguns instrumentos de visualização de informação em uma perspectiva multidimensional para auxiliar professores na avaliação diagnóstica da aprendizagem de programação com mapeamento de perfis em métricas de software. Através das visualizações geradas, é possível analisar e comparar perfis sob diferentes variáveis, evidenciar dificuldades de aprendizagem e identificar classes de soluções a partir de características semelhantes.

A estratégia de reconhecimento de perfis por métricas de análise estática de códigos de [Novais et al. 2016] tem como objetivos inferir perfis de programadores a partir da análise de seus códigos em Java, classificá-los conforme habilidades e avaliar continuamente seus progressos na prática da programação em um curso. Os perfis detectados são noviço, iniciante avançado, proficiente e especialista. Algumas das métricas utilizadas são: número de sentenças, de estruturas de controle condicional e de repetição, de tipos de dados, de classes, de operadores, de linhas de código e outras informações de código. A vantagem dessa estratégia em relação ao sistema deste trabalho é utilizar as métricas para classificar e qualificar alunos. No entanto, selecionamos automaticamente as métricas mais adequadas para avaliar cada tipo de solução de programação.

Para seleção automática de variáveis de avaliação, destacamos o modelo de seleção de características de [Spalenza et al. 2016], que combina as técnicas de *clustering* e de algoritmo genético para criar um mapa de características selecionando termos relevantes nos textos dentre os grupos de notas da avaliação de um professor. Em nossa proposta, as características relevantes, isto é, as métricas mais importantes para cada solução de programação, podemos visualizar através de mapas de calor comparando diferentes soluções de programação por cinco ou mais métricas de software.

Em relação à composição de rúbricas, uma estratégia a se destacar é a proposta de [Gonçalves de Oliveira et al. 2018], que é baseada em técnicas de *clustering* e de *Análise de Componentes Principais* para reconhecer, a partir de soluções desenvolvidas por alunos, exemplos de soluções que representem, em um esquema de rubricas, os escores atribuídos por um professor. Um outro exemplo que estende essa proposta é o trabalho de [Olmos et al. 2016], que utiliza redução de dimensionalidade por meio da técnica *Latent Semantic Analysis (LSA)* na avaliação de documentos textuais para compor rubricas a partir de conceitos visando reconhecer e avaliar eixos conceituais de um texto. O sistema deste trabalho complementa essas propostas ao gerar um *ranking* de amostras de soluções de programação para um professor pontuar até encontrar o melhor conjunto de representações de rubricas com uma diversidade de notas atribuídas.

De acordo com [Tang et al.], para entender como a aprendizagem se desdobra ao longo do tempo, é necessário mudar para uma nova perspectiva de aprendizagem em que as unidades de análise são eventos de aprendizagem separados, mas inter-relacionados. Seguindo essa ideia, o estudo de [Tang et al.] investiga e valida padrões longitudinais na participação *online* como uma medida para diferenciar desempenhos de estudantes.

A proposta do sistema deste trabalho, baseando-se no estudo de [Tang et al.], busca compreender como a aprendizagem de programação se desdobra e analisar padrões longitudinais. Dessa forma, seguindo essa proposta, em relação aos demais trabalhos apresentados, avançamos na representação 3D de perfis de estudantes de programação, na visualização de características representadas por métricas de software ao longo do tempo e na composição de rubricas a partir de um *ranking* de soluções selecionadas automaticamente para um professor pontuar.

3. Sistema de Representação 3D de Perfis de Estudantes de Programação

O sistema de representação de perfis apresentado neste trabalho é uma evolução do software *PCodigo II* desenvolvido por [Neves et al. 2017] para análise de dificuldades de aprendizagem, boas práticas de programação e plágios por meio de métricas de software que quantificam esforço e qualidade de programação. O sistema deste trabalho estende a representação de perfis do *PCodigo II* em uma dimensão temporal, seleciona métricas mais relevantes e possibilita a seleção automática de exemplos representativos de códigos-fontes para composição de representações de rubricas de exercícios de programação.

A Figura 1 apresenta a arquitetura do sistema proposto neste trabalho apresentando as entradas, as funcionalidades e as saídas geradas.

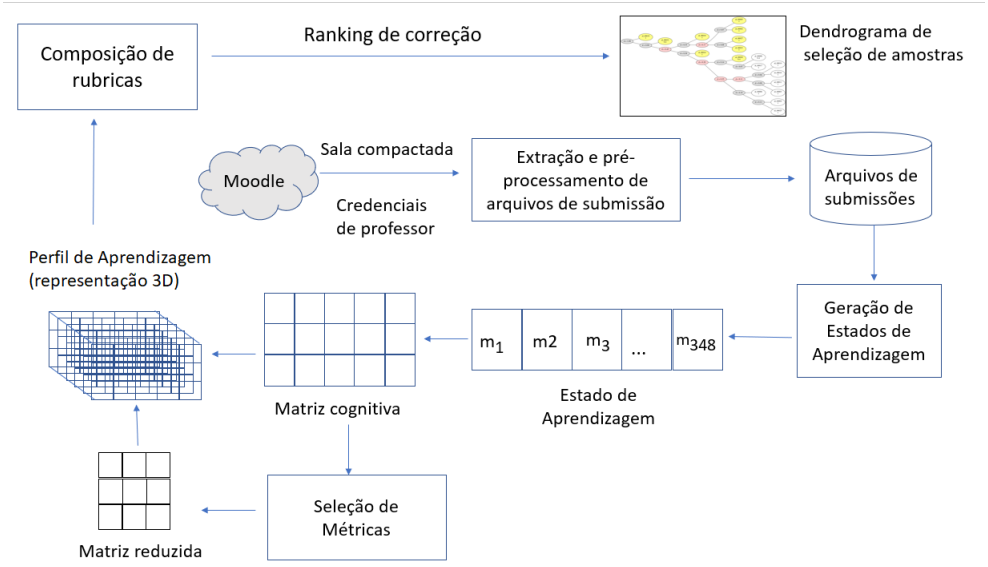

Figura 1. Arquitetura do sistema de representação de perfis 3D

O sistema representado na Figura 1 foi desenvolvido com integração às versões 1.0 e 3.x do *Moodle*. Para a versão 1.9 do *Moodle*, o sistema recebe como entrada um *backup* de uma sala de curso de programação compactada (em *.zip, .rar, .gz ou .tgz*). Para a versão 3.x do *Moodle*, a entrada do sistema vem dos dados de uma sala de curso de programação obtidos de um servidor remoto do *Moodle* a partir das credenciais de acesso de um professor a um curso de programação a distância. Os dados de curso importados do *Moodle* são os seguintes: a listagem de alunos, a listagem de atividades, as notas das atividades e os arquivos de submissões de exercícios de programação.

Esses dados são então extraídos pelo sistema e as submissões que contêm códigos-fontes escritos em Linguagem C, C++, Java ou *Python* são mapeados em vetores cujas dimensões são métricas de software que quantificam esforço e qualidade de programação [Neves et al. 2017]. As submissões em C, C++ e Java são mapeados nas 348 métricas de software de [Neves et al. 2017] e as submissões em *Python*, em 42 métricas.

Neste trabalho chamamos de *Estado de Aprendizagem* a representação vetorial em métricas de software da submissão de um estudante.

Após gerar os *Estados de Aprendizagem* de cada estudante para um exercício de programação, o sistema reúne essas representações em uma *Matriz Cognitiva* para análise e comparação das soluções dos estudantes [Neves et al. 2017].

Para analisar as soluções de forma genérica, reduzimos um *Estado de Aprendizagem* a cinco métricas: *Manutenibilidade, Complexidade Ciclomática, Indentação, Laconismo* e *Modularização*.

A *Manutenibilidade* representa a capacidade do aluno de escrever código durável, adaptável a novas necessidades. A *Complexidade Ciclomática* informa a complexidade de um código de programação que é o número de caminhos de um método [Curtis et al. 1979]. A *Indentação* caracteriza organização das instruções de um programa dentro de estruturas e funções. O *Laconismo* expressa a capacidade de expressar-se em poucas palavras e, em programação, é medido pelo número de *tokens* por linha de código. Já a *Modularização* informa a capacidade de organização das partes de um programa em módulos funcionais ou de dados.

Após a representação dos *Estados de Aprendizagem* em cinco métricas, é gerada uma *Matriz Cognitiva Reduzida*.

Em seguida, reunindo as matrizes cognitivas para cada exercício de programação de um curso com submissão de códigos-fontes, gera-se uma representação 3D de todos os *Estados de Aprendizagem* dos estudantes de uma turma ao longo de um curso. O mesmo procedimento é realizado para as matrizes cognitivas reduzidas.

A representação do conjunto dos *Estados de Aprendizagem* de um estudante em cada exercício de programação de um curso, isto é, em uma linha de tempo, chamamos de *Perfil de Aprendizagem*. O *Perfil de Aprendizagem* indica como as variáveis de avaliação de um estudante evoluem ao longo de um curso. Dessa forma, através da análise de perfis de aprendizagem, é possível compreender as principais dificuldades de aprendizagem dos estudantes e reorientar o ensino com ações de avaliação formativa de forma a antecipar-se a um futuro previsto de fracasso acadêmico.

3.1. Seleção de códigos e características para composição de rubricas

Para a composição de representações de rubricas com o propósito de auxiliar professores nas decisões de avaliação de exercícios de programação, desenvolvemos uma estratégia de seleção automática de amostras representativas de soluções de exercícios de programação e de características mais relacionadas com as notas atribuidas por um professor a esse pequeno conjunto de amostras representativas.

Para a seleção dessas amostras, utilizamos uma representação hierárquica de *clusters* em dendrograma com medida de similaridade de *Distância Euclidiana*. Através dessa

representação, por busca em profundidade (BFS) não ciente de plágio, onde a profundidade é dada pelas distâncias acumuladas (da raiz até o nó) expressas nos nós.

Depois que as amostras selecionadas são pontuadas pelo professor, verificam-se quais características mais impactam nas notas atribuídas por um professor.

4. Experimentos e resultados

Os primeiros experimentos das funcionalidades do sistema proposto neste trabalho foram realizados em uma sala *Moodle* de um curso de programação C a distância do Centro de Referência em Formação e Educação a Distância do Instituto Federal do Espírito Santo (Cefor/Ifes). Através das credenciais de acesso de um professor de programação, obtivemos uma cópia em arquivo zipado da sala desse curso de programação a distância para início de processamento do nosso sistema de avaliação.

Em seguida, foram extraídos todos os arquivos de códigos de programação C submetidos ao longo do curso por cerca de 25 estudantes de programação.

Após a geração das representações 3D de perfis de aprendizagem (Atividades X Alunos x Métricas) reunindo dez atividades, 25 alunos e 348 métricas de software, utilizamos essas informações para gerar as seguintes resultados e visualizações:

1. Para cada atividade, a lista de métricas que foram consideradas as mais relevantes para a atribuição de nota;
2. Para a turma como um todo, a lista de métricas que foram consideradas as mais relevantes para a atribuição de nota;
3. Dendrograma gerado automaticamente sobre todas as métricas que compõem o perfil de aluno;
4. Dendrograma gerado automaticamente sobre todas as métricas que compõem o perfil de aluno, após normalização a valores entre 0 e 1;
5. Mapa de calor para cada atividade com as métricas selecionadas que melhor representam cada atividade, individualmente;
6. Mapa de calor para cada atividade com as métricas que melhor representam o critério de correção para a turma como um todo;
7. Mapa de calor para cada aluno (histórico no tempo) com as métricas que melhor representam o critério de correção para a turma como um todo;
8. Mapa de calor para cada atividade com cinco métricas que representam as habilidades e dificuldades de programação do aluno;
9. Mapa de calor para cada aluno (histórico no tempo) com cinco métricas que representam as habilidades e dificuldades de programação do aluno;
10. Predição de notas de alunos, onde as notas são atribuídas a submissões que são similares entre si.

Neste trabalho, apresentamos os resultados 1, 5, 8, 9 e 10. Os resultados 1, 5 e 8 aparecem na Figura 2; o resultado 9, na Figura 3 e o resultado 10, na Figura 4.

Na Figura 2, destacamos dois modelos de análise de soluções de programação para uma atividade de programação a partir de métricas de software: a partir das métricas *Manutenibilidade, Complexidade Ciclomática, Indentação, Laconismo e Modularização* e a partir de métricas que foram consideradas as mais relevantes para a atribuição de notas, isto é, as métricas da *Matriz reduzida*.

Nos gráficos das figuras 2 e 3, as colunas indicam os alunos e as colunas, as métricas. Em cada coluna, a cor branca (escala 1) sinaliza o maior valor e a cor preta (escala 0), o menor valor de uma métrica. A interpretação se é melhor o valor mais alto ou baixo depende do nível de informações de cada código de programação. Mas o professor pode realizar essa interpretação comparando valores das melhores soluções e das poiores soluções. Dessa forma, ele teria um instrumento para avaliar que indicadores caracterizam as boas soluções de programação e aquelas que expressam mais dificuldades.

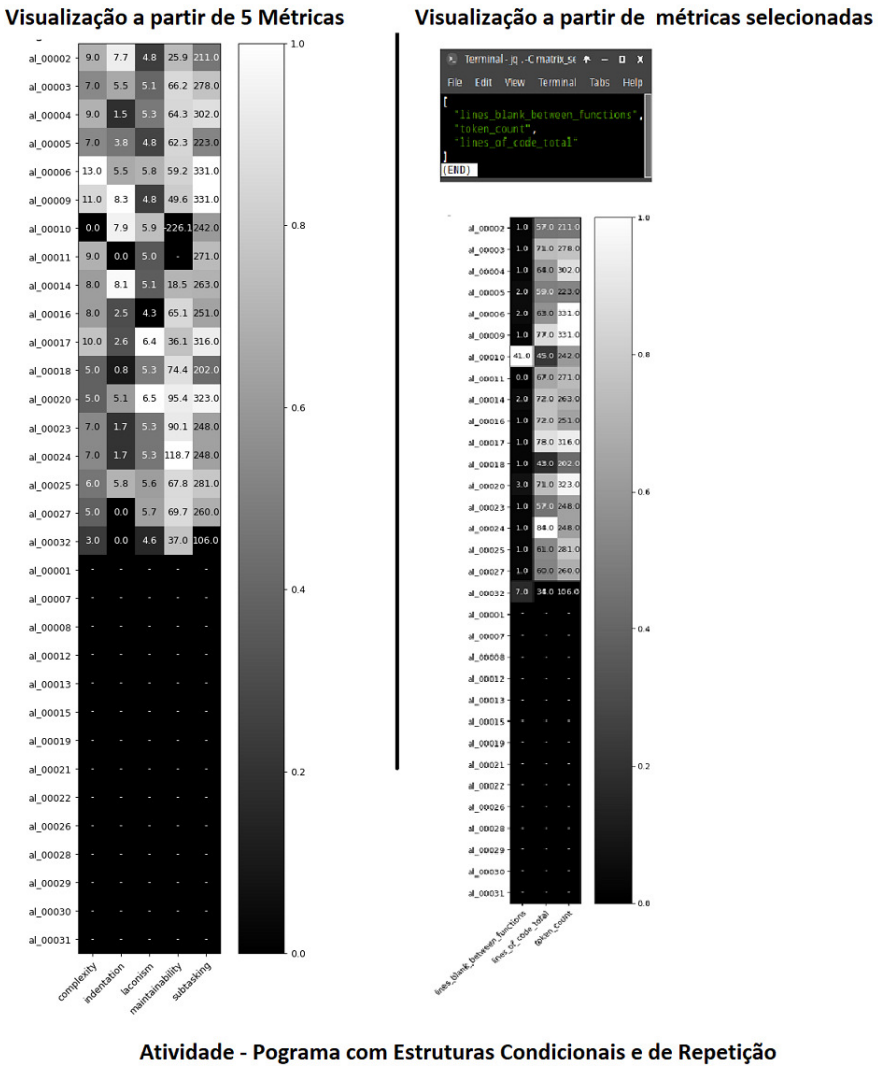

Figura 2. Análise de soluções por métricas de software

De acordo com a Figura 2, no primeiro gráfico, as métricas *Complexidade*, *Manutenibilidade* (valor baixo) e *Identação* (Valor alto) diferenciam a solução do *Aluno 10* das demais e destacam-no como um bom programador. Isso pode ser confirmado no segundo gráfico no valor alto da primeira métrica (espaços em branco entre as funções) e no valor baixo na segunda coluna, que é a métrica do número de linhas de código.

Na Figura 3, destacamos como as cinco principais métricas evoluem a cada exercício para um mesmo aluno. Observa-se que esse aluno, no primeiro exercício (linha

1), apresenta predomínio da cor preta, indicando baixos valores, indicando bons desempenhos nos exercícios mais fáceis. Por outro lado, no último exercício que ele fez, as cores aparecem mais claras, sinalizando atividades mais complexas e mais dificuldades. Isso fica mais evidente quando, a partir desse exercício, o aluno parou de entregar as atividades de programação. Vemos nessa visualização o potencial da ferramenta de informar em que ponto um aluno começa a apresentar dificuldades, que é onde o professor pode começar a intervir de forma a recuperar esse aluno.

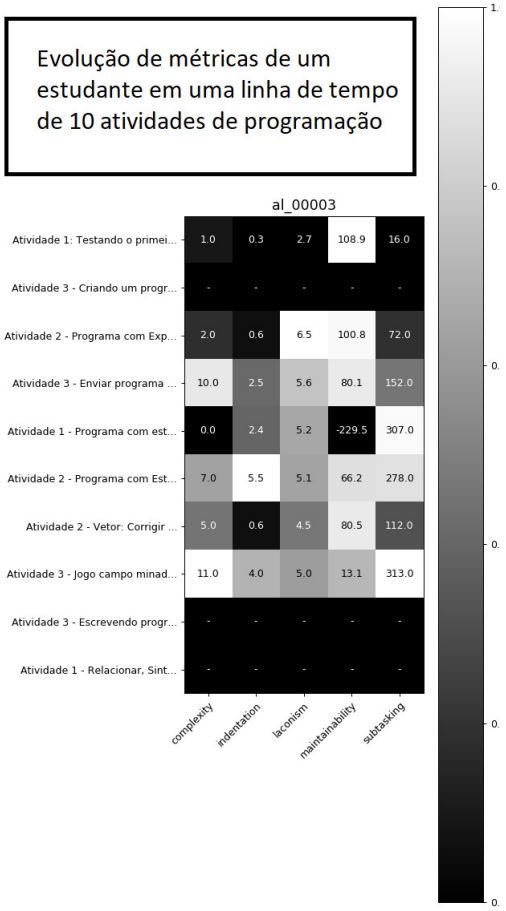

Figura 3. Linha do tempo de um estudante de programação

O gráfico da Figura 4 é a visualização do *ranking* de amostras selecionadas (em amarelo) automaticamente pelo sistema para o professor pontuar com menor esforço de correção. O gráfico é um dendrograma que apresenta a hierarquia das soluções de desenvolvidas para uma atividade de programação. As distâncias são marcadas em cinza e rosa. O critério de seleção das amostras foi selecionar primeiro as amostras de maior distância euclidiana, isto é, de maior dissimilaridade.

Conforme a Figura 4, selecionando primeiro as amostras de maior dissimilaridade, o professor pontua aquelas mais diferentes e, em seguida, algumas das mais similares. À medida que ele segue o *ranking* de amostras sugerido pelo sistema, ele próprio pode

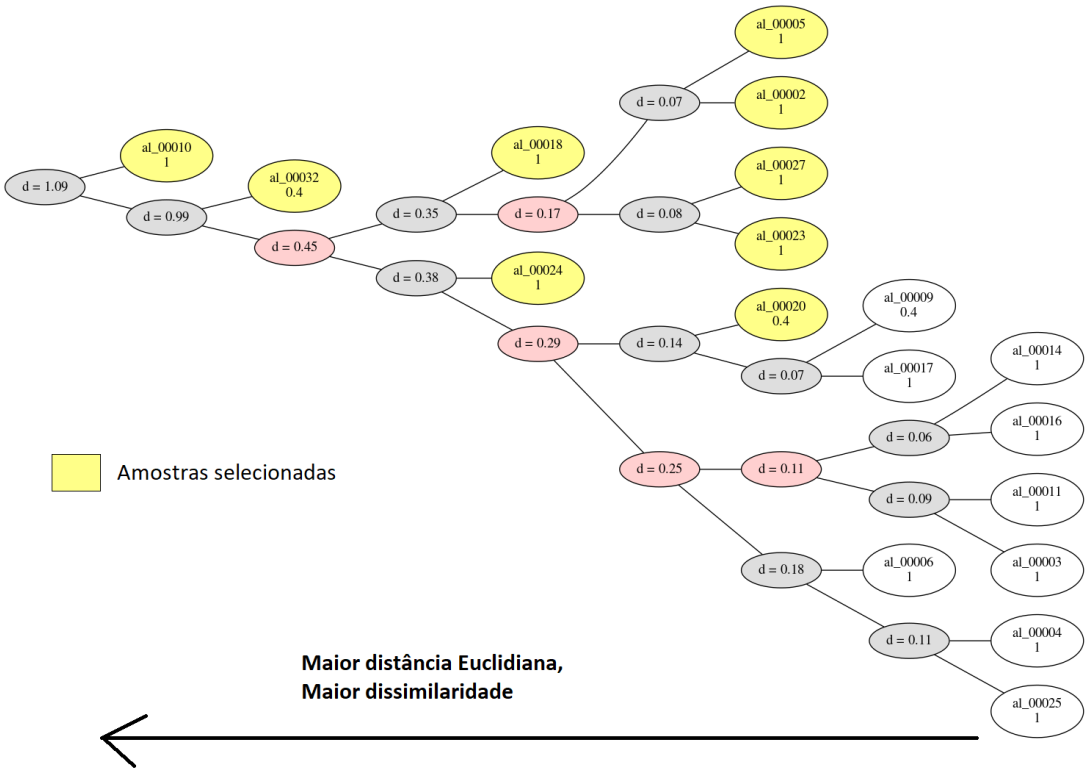

Figura 4. Seleção de soluções de programação para composição de rubricas

identificar até onde poderá corrigir para obter um conjunto mínimo de representação da diversidade das soluções desenvolvidas para composição de rubricas e, futuramente, para treinar sistemas de avaliação automática de exercícios de programação com um conjunto representativo de exemplos de notas.

Na Figura 4, observamos que as primeiras amostras indicadas para correção, que estão em cor amarela, já representam o conjunto de notas possíveis nesse exercício de programação: 1 (mais alta) e 0.4 (mais baixa).

Concluindo, com alguns exemplos dos resultados gerados pelo sistema deste trabalho, mostramos o potencial dessa ferramenta para professores de programação acompanharem o processo de aprendizagem de seus alunos do início ao fim de um curso a partir de um conjunto amplo ou reduzido de métricas e com menor esforço de avaliação.

5. Considerações finais

O sistema proposto neste trabalho apresentou-se como uma relevante ferramenta para auxiliar professores nas decisões de um processo avaliativo de programação possibilitando de fato assistir a aprendizagem de estudantes de programação a cada exercício favorecendo a identificação de onde começam as dificuldades de aprendizagem, o acompanhamento de como uma turma evolui ao longo de um curso e a composição dinâmica de representações de rubricas, com menor esforço de avaliação de professores.

Os trabalhos futuros a partir deste trabalho são utilizar as amostras indicadas para correção como referências de treino de um sistema de avaliação semi-automática de programação e desenvolver uma estratégia para previsão de desempenhos em atividade-

des de testes e provas a partir da linha do tempo de exercícios de programação resolvidos ou a partir de soluções de alunos que resolveram exercícios similares àquele que visamos predizer uma nota.

Através deste trabalho oferecemos, portanto, uma ferramenta para ajudar professores em suas ações formativas e estudantes a serem melhor assistidos em suas dificuldades e habilidades na prática da programação.

6. Agradecimentos

Agradecemos à Fundação de Apoio à Pesquisa e Inovação do Espírito Santo (FAPES) e ao Ifes pelo apoio dado ao projeto Tecnologias de Avaliação Semi-automática da Aprendizagem de Programação (do EDITAL 006/2014 – Universal Projeto Individual de Pesquisa) do qual resultou esta publicação.

Referências

Curtis, B., Sheppard, S. B., Milliman, P., Borst, M., and Love, T. (1979). Measuring the psychological complexity of software maintenance tasks with the halstead and mccabe metrics. *IEEE Transactions on software engineering*, (2):96–104.

De Oliveira, M. G., Ciarelli, P. M., and Oliveira, E. (2013). Recommendation of programming activities by multi-label classification for a formative assessment of students. *Expert Systems with Applications*, 40(16):6641–6651.

Gonçalves de Oliveira, M., Batista de Souza, M., Leal Reblin, L., and Silva de Oliveira, E. (2018). Reconhecimento automático de representações de rubricas em agrupamentos de soluções de exercícios de programação. *Revista Brasileira de Informática na Educação*, 26(2).

Neves, A., Reblin, L., França, H., Lopes, M., Oliveira, M., and Oliveira, E. (2017). Mapeamento automático de perfis de estudantes em métricas de software para análise de aprendizagem de programação. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 1337.

Novais, D., Pereira, M. J., and Henriques, P. R. (2016). Profile detection through source code static analysis. In *5th Symposium on Languages, Applications and Technologies (SLATE'16)*, volume 51, pages 1–13. OASICS.

Oliveira, M., França, H., Neves, A., Lopes, M., and Silva, A. C. (2017). Instrumentos de visualização da informação para avaliação diagnóstica em curso de programação a distância. In *Anais do Workshop de Informática na Escola*, volume 23, page 452.

Olmos, R., Guillermo, J. B., Luzón, J. M., Martín-Cordero, J. I., and Leao, J. A. (2016). Transforming LSA space dimensions into a rubric for an automatic assessment and feedback system. *Information Processing & Management*, 52(3):359 – 373.

Spalenza, M., Oliveira, E., Oliveira, M., and Nogueira, M. (2016). Uso de mapa de características na avaliação de textos curtos nos ambientes virtuais de aprendizagem. In *Anais do SBIE 2016*, pages 1165–1174.

Tang, H., Xing, W., and Pei, B. Time Really Matters: Understanding the Temporal Dimension of Online Learning Using Educational Data Mining. *Journal of Educational Computing Research*, 0(0):0735633118784705.