Análise de Componentes Latentes da Aprendizagem de Programação para Mapeamento e Classificação de Perfis

Márcia G. de Oliveira [1], Nátaly A. Jiménez Monroy [2], Eliana Zandonade [2], Elias Oliveira [1]

[1] Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo
Vitória – ES – Brasil

[2]Departamento de Estatística
Universidade Federal do Espírito Santo
Vitória – ES – Brasil

clickmarcia@gmail.com, elias@acm.org

Abstract. *This work aims to recognize, from programs developed by students and from a network of variables that represents the field of programming some latent components that characterize the performance of a class in each programming activity. The latent components identification is performed by factor analysis to represent students profiles in the practice of programming. By clustering, we form clusters of similar profiles. Then we selected pre-classified samples from each formed cluster to generate models to guide the classification of other profiles according to their learning levels. The results indicate the combination of factorial analysis and clustering techniques improve the classification of profiles.*

Keywords: *Factorial analysis, clustering, learning programming, mapping of profiles, classification of profiles.*

Resumo. *Este trabalho tem como objetivo reconhecer, a partir de programas desenvolvidos por alunos e de uma rede de variáveis representantes do domínio da programação, componentes latentes que caracterizem os desempenhos de uma turma em cada atividade de programação. A identificação das componentes latentes é realizada pela técnica de análise fatorial para representar perfis de aprendizagem na prática da programação e, por clustering, formam-se agrupamentos de perfis similares. De cada agrupamento formado, são selecionadas amostras de perfis pré-classificadas para gerar modelos que orientem a classificação dos demais perfis de cada agrupamento por níveis de aprendizagem. Os resultados indicam que a combinação das técnicas de análise fatorial e de clustering melhora a classificação de perfis.*

Palavras-chave: *Análise fatorial, Clustering, Aprendizagem de programação, Mapeamento de perfis, Classificação de perfis*

1. Introdução

A programação de computadores é o processo de escrever em linguagem formal instruções sequenciadas logicamente com o propósito de resolver um problema através de uma

solução automatizada. Mas, por envolver a combinação de várias habilidades cognitivas, a prática da programação é complexa. Para se ter uma ideia da complexidade de programar, para desenvolver um programa de computador, são necessárias as seguintes atividades cognitivas [Pea and Kurland 1984]:

1. Compreender o problema
2. Planejar ou projetar uma solução
3. Escrever código do plano de solução
4. Compreender a escrita do programa
5. Depurar o programa

Para contemplar a aprendizagem de programação, hoje existem sofisticadas tecnologias para apresentar conteúdos de programação [Powers et al. 2006], para instruir [Pillay 2003], para predizer comportamentos [Mavrikis 2010], para dar instruções e exercícios personalizados de acordo com os perfis de alunos [Mazza and Dimitrova 2007], para recomendar atividades [De Oliveira et al. 2013] e até para avaliação automática de exercícios de programação [Moreira and Favero 2009, Naude et al. 2010]. Mas carecemos de tecnologias que de fato ofereçam uma prática assistida, isto é, planejada, monitorada, controlada e com *feedback* imediato no domínio da aprendizagem de programação.

Para melhor assistir e regular a prática da programação, no entanto, é necessário conhecer continuamente estados de aprendizagem de alunos representando-os através de perfis que sejam mapeados em diferentes variáveis de avaliação que apontem suas habilidades e dificuldades de aprendizagem e os ajudem na remediação da aprendizagem [Anderson 2000, Mazza and Dimitrova 2007, De Oliveira et al. 2013]. Caminhando nessa direção, o trabalho de [Oliveira and Oliveira 2014] chama a atenção para a representação de perfis por componentes de habilidades que representem desempenhos em variáveis de avaliação do domínio da aprendizagem de programação.

Aplicando essa representação, [Oliveira and Oliveira 2014] propõem um Núcleo de Avaliação Diagnóstica (NAD) que reúne as funções de mapeamento de perfis e de correção semiautomática de exercícios de programação. No NAD, o mapeamento de perfis é realizado a partir dos desempenhos dos alunos em componentes de habilidades que representam o domínio de aprendizagem da programação [Oliveira and Oliveira 2014]. Dessa forma, uma turma é representada por uma matriz cognitiva A_{mxn}, em que m representa o número de alunos e n, o número de componentes de habilidades.

Os desempenhos mapeados em componentes de habilidades são obtidos por tarefas (listas de exercícios e provas), onde cada atividade requer como resposta do aluno um programa de computador desenvolvido em Linguagem C [Oliveira and Oliveira 2014]. No programa submetido, a frequência de ocorrência das palavras reservadas, dos operadores e dos símbolos bem como os indicadores de execução constituem as medidas das componentes de habilidades que representam um perfil de aluno.

Este trabalho estende o Núcleo de Avaliação Diagnóstica de [Oliveira and Oliveira 2014] ao propor uma redução de dimensionalidade da matriz A através da técnica de análise fatorial. Essa nova matriz é representada pelas componentes latentes presentes nas relações entre as componentes de habilidades da matriz A. Através dessa nova representação, objetiva-se promover melhoramentos nos processos de classificação de perfis.

Os resultados de experimentação deste trabalho indicam que a combinação das técnicas de análise fatorial e de *clustering* melhoram a classificação de perfis. A contribuição desta proposta para o domínio da aprendizagem de programação é, portanto, oferecer um mecanismo de seleção de características e amostras de perfis que melhor representem a prática de programação e possibilitem a classificação de novos perfis conforme seus níveis de aprendizagem.

As seções deste artigo estão organizadas conforme a ordem a seguir. Na Seção 2, as técnicas de análise fatorial e de *clustering* e alguns trabalhos relacionados são explicados. Na Seção 3, é apresentada a estratégia de mapeamento e de classificação de perfis deste trabalho. Na Seção 4, são discutidos os experimentos realizados e os resultados alcançados. Na Seção 5, conclui-se este trabalho com as considerações finais.

2. Fundamentação Teórica

A avaliação diagnóstica, como instrumento de coleta e análise de informações de um processo de aprendizagem, deve ser utilizada para informar, através de variáveis de avaliação, o estado de aprendizagem dos alunos. As informações obtidas a partir da avaliação diagnóstica devem, portanto, permitir a exploração do conhecimento e sintetizá-lo em um perfil ou modelo do aluno.

Dessa forma, um processo de aprendizagem pode ser sintetizado em uma matriz cognitiva [Mazza and Dimitrova 2007] que mapeie as aprendizagens dos alunos de uma turma em variáveis de avaliação e classes de perfis [Oliveira and Oliveira 2008]. Essa matriz deve informar grupos de alunos com perfis de aprendizagens similares e auxiliar professores na instrução adaptativa [Pimentel et al. 2003].

2.1. A Análise Fatorial

A análise fatorial é uma técnica de análise multivariada que consiste na redução da dimensionalidade de um conjunto de dados, encontrando grupos homogêneos de variáveis a partir de um grande número de variáveis. Esses grupos homogêneos, também chamados de *fatores*, são formados por variáveis muito correlacionadas entre si. O ideal é que os fatores sejam independentes uns dos outros. Portanto, a análise fatorial é uma técnica que busca explicar ao máximo a informação contida nos dados, usando a menor quantidade possível de dimensões.

De acordo com [Härdle and Simar 2012], a análise é baseada em um modelo onde o vetor observado é representado por uma parte sistemática e uma parte de erro não observado. As componentes do vetor de erros são consideradas não-correlacionadas ou independentes, enquanto que a parte sistemática é considerada como uma combinação linear de um número relativamente pequeno de fatores não observados.

Frequentemente as influências dos k fatores costumam ser divididas em *comuns* e *específicas*. Por exemplo, há fatores altamente informativos que são comuns para todas as p componentes de $\mathbf{X}$ e fatores que são específicos apenas para algumas componentes. Nesse caso, a matriz $\mathbf{X}$ pode expressar-se como

$$\mathbf{X} = \mathbf{QF} + \mathbf{U} + \mu,$$

onde $\mathbf{Q}$ é uma matriz $(p \times k)$ de coeficientes (não-aleatórios) dos fatores comuns $\mathbf{F}$ (de dimensão $(k \times 1)$), $\mathbf{U}$ é uma matriz $(p \times 1)$ dos fatores (aleatórios) específicos e μ é o vetor de médias de $\mathbf{X}$.

Assumindo que um modelo fatorial com k fatores foi encontrado razoável, isto é, a maioria das variações das p variáveis em $\mathbf{X}$ foram explicadas por k fatores fixos, podemos interpretar os fatores F_l, $l = 1,\ldots,k$ calculando suas correlações com as variáveis X_j, $j = 1,\ldots,p$, para obter a matriz $\mathbf{P}_{XF}$. Essa correlação é dada por

$$P_{XF} = D^{-1/2}Q,$$

onde $D = diag\{\sigma_{X_1X_1},\ldots,\sigma_{X_pX_p}\}$, sendo $\sigma_{X_jX_j}$ a variância de X_j. Usando essa informação podemos determinar quais das variáveis originais $X_1,\ldots,X_p$ influenciam nos fatores não observados $F_1,\ldots,F_k$ [Anderson 1984, Bishop et al. 1975, Morrison 1990].

2.2. O Clustering

O *clustering* é uma técnica que utiliza a abordagem de aprendizagem não-supervisionada para agrupamento de padrões em classes ou *clusters* considerando as características semelhantes desses padrões. O objetivo do *clustering* é formar grupos caracterizados por alta homogeneidade entre padrões de um mesmo grupo e heterogeneidade entre padrões de grupos distintos.

O *Bisecting K-means* é uma variação do algoritmo *K-means*. Ele começa com um simples *cluster* e continuamente seleciona um *cluster* para dividir em dois *sub-clusters* até alcançar o número *K* de *clusters* desejados [Looks et al. 2007].

De acordo com [Steinbach et al. 2000], o *Bisecting K-means* tem apresentado melhor performance do que o *K-means* em muitos casos devido ao fato de produzir *clusters* de tamanhos mais uniformes em vez de *clusters* de tamanhos variáveis.

2.3. Trabalhos Relacionados

As técnicas de análise multivariada como a análise fatorial e o *clustering* foram aplicadas para avaliação da aprendizagem em diferentes domínios do conhecimento com o objetivo de identificar fatores que caracterizem um domínio e para reconhecimento de classes de perfis de aprendizagem.

O trabalho de [Law et al. 2010], por exemplo, por análise fatorial, apresenta um estudo preliminar que investiga os fatores-chave de motivação que afetam a aprendizagem entre estudantes de cursos de programação. Esses cursos são suportados por um sistema *online*, o *PASS (Programming Assignment Assessment System)*, que visa fornecer uma infraestrutura para facilitar a aprendizagem de programação.

No artigo de [Blikstein 2011], é descrita uma técnica automatizada para avaliar, analisar e visualizar os estudantes que aprendem programação de computadores. Assim, registram-se centenas de códigos dos alunos e empregam-se diferentes técnicas quantitativas para extrair os comportamentos dos alunos e categorizá-los em termos de suas experiências em programação.

Em um trabalho mais recente, [Oliveira 2013] mapeia perfis em componentes de habilidades e aplica a técnica de *clustering* para classificar esses perfis. Além disso, [Oliveira 2013] mostra como algoritmos de *clustering* melhoram a seleção de características e amostras para composição de modelos de regressão utilizados na avaliação semiautomática de exercícios de programação.

Assim como [Mesic and Muratovic 2011] selecionam fatores que influenciam a dificuldade de itens de Física para identificação de preditores de modelos de regressão, propomos estender o trabalho de [Oliveira and Oliveira 2014] utilizando a análise fatorial para reduzir o número de componentes de habilidades a um número de componentes latentes para representar e classificar perfis.

3. Mapeamento e Classificação de Perfis

A estratégia de mapeamento e classificação de perfis deste trabalho é apresentada na Figura 1. O módulo de mapeamento de perfis recebe programas de computador em Linguagem C e os transforma em vetores cujas dimensões são chamadas de componentes de habilidades (C_i). Cada componente de habilidade é a frequência de ocorrência de palavras reservadas, símbolos, operadores e funções da Linguagem C ou indicadores de funcionamento como a compilação e a execução de programas.

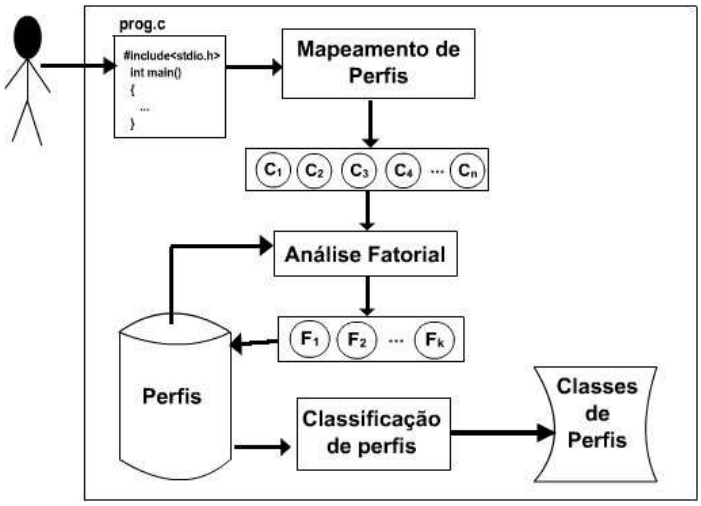

Figura 1. Mapeamento e classificação de perfis

O módulo de análise fatorial, conforme a Figura 1, redimensiona os perfis gerados pelo módulo de mapeamento de perfis extraindo as componentes latentes através das relações entre as componentes de habilidades C_i. Em seguida, os perfis, agora representados por fatores F_j representando as componentes latentes, são submetidos ao algoritmo de *clustering Bisecting K-means* do módulo de classificação de perfis e são agrupados conforme as suas similaridades.

De cada *cluster*, são selecionadas 2/3 de amostras aleatórias de perfis para gerar modelos de regressão para predizer *scores* indicadores de níveis de aprendizagem. Em seguida, os *scores* dos demais 1/3 dos perfis de cada *cluster* são preditos e uma classe de nível de apredizagem é associada a cada um desses perfis.

O objetivo dessa metodologia é mostrar como a combinação das técnicas de análise fatorial e de *clustering* pode melhorar o processo de classificação de perfis.

4. Experimentos e Resultados

A metodologia experimental deste trabalho foi realizada conforme os seguintes passos:

1. Seleção de 100 amostras de programas em Linguagem C desenvolvidos por alunos de programação para uma atividade
2. Mapeamento das amostras de programas em vetores reunidos em uma matriz A
3. Análise multivariada fatorial da matriz A
4. Clusterização da matriz A
5. Separação dos conjuntos de treino e de teste de cada *cluster*
6. Criação de modelos de regressão linear para cada *cluster* a partir do conjunto de treino pré-classificado
7. Predição de *scores* do conjunto de teste de cada *cluster*
8. Classificação das amostras do conjunto de teste de cada *cluster*
9. Avaliação de resultados

Os 100 programas reunidos em uma base chamada *Base-p* foram representados por uma matriz A_{mxn}, onde *m* é o número de linhas representadas pelos 100 programas desenvolvidos por alunos e *n*, o número de colunas representando as 60 componentes de habilidades, que são variáveis cujos valores são a frequência de ocorrência de palavras reservadas, operadores, símbolos e funções da Linguagem C. As varíaveis de execução indicando se um programa compilou e executou assumem valores 0 e 1, que significam, respectivamente, *falso* e *verdadeiro*. A Tabela 1 mostra como os programas desenvolvidos por alunos foram vetorialmente representados na matriz *A*. Cada linha dessa matriz corresponde a um perfil de aluno mapeado a partir dos desempenhos desse aluno em cada componente de habilidade.

100	60																			
1	1	3	0	0	0	0	0	9	2	0	0	0	0	0	2					
1	1	0	0	0	0	0	0	9	0	0	0	0	0	0	2					
0	0	0	0	1	0	0	0	9	0	0	0	0	0	0	2					
1	1	5	4	2	0	1	0	7	0	0	0	0	0	0	2					
1	1	1	3	0	0	0	0	2	1	0	0	0	0	0	1					
1	1	0	0	0	0	0	0	4	4	0	0	0	0	0	2					
1	1	0	0	0	0	0	0	6	5	0	0	0	0	0	2					

Tabela 1. Representação vetorial de programas em Linguagem C

Após a geração da Matriz *A*, foi realizada a análise multivariada fatorial com rotação *Varimax* para agrupar as variáveis correlacionadas e criar possíveis fatores representantes da atividade de programação aplicada nas turmas de Engenharia e Computação. O programa estatístico utilizado foi o *SPSS*, versão 18.0.

Das 60 variáveis iniciais restaram para análise apenas 35, pois as demais eram somente zero. Dessa análise, consideraram-se doze fatores, sendo que o percentual de variabilidade explicado por eles foi de 82.09%. Adotou-se como critério de decisão do número de fatores autovalores maiores que 1. Os autovalores representam a variabilidade explicada por cada fator.

A Figura 2 apresenta as cargas fatoriais de quatro dos doze fatores (ou componentes latentes) do modelo gerado. Nota-se que as variáveis agrupadas em diferentes cores definem cada fator.

Na Tabela 2 são apresentados os doze fatores selecionados para caracterizar o exercício de programação resolvido pelos 100 alunos. As características principais são representadas por identificadores de palavras-reservadas (*cbreak, ccase,cswitch, cfor, tchar,*

Nome do fator	1	2	3	4	5	6	7	8	9	10	11	12
opmais	,904	,005	-,008	-,053	,203	-,052	-,044	,027	,154	,126	,009	,020
opmult	,876	,026	-,149	-,040	,287	-,069	-,071	,027	,173	,136	-,029	,026
opmenos	,848	-,016	-,055	,007	,199	,110	-,048	,020	-,128	,087	,090	,152
endereco	,774	-,098	-,291	-,017	-,064	,015	,010	,121	,206	-,247	-,085	-,095
cimprimir	,618	-,010	-,159	-,041	,285	,006	,186	,197	,076	,061	,024	-,256
cbreak	,004	,974	,094	,037	,046	-,035	,038	,036	,009	-,030	,017	,025
ccase	-,012	,986	-,015	-,015	,039	-,012	,023	,009	,008	-,022	,060	-,006
cswitch	-,012	,986	-,015	-,015	,039	-,012	,023	,009	,008	-,022	,060	-,006
tvoid	-,199	,487	,072	-,065	,095	,390	-,016	,153	,009	,014	,026	-,480
opmenor	,254	-,082	,432	,171	,201	,077	-,015	-,047	-,011	,333	-,323	-,288
opdec	-,002	-,015	,809	,174	-,040	-,108	,133	,095	,038	-,079	,018	-,010
creturn	-,004	,227	,571	,304	,008	-,071	-,082	,573	,054	,019	,053	-,175
cfor	-,285	,028	,807	-,065	-,174	,183	-,129	-,049	,040	,037	-,017	-,001
opinc	-,293	,071	,851	,046	-,079	,160	-,128	-,022	,035	-,022	-,110	,133
tchar	,046	-,016	,351	,817	,127	-,084	-,012	,044	,075	,035	-,229	,069
opor	,024	-,037	-,177	,853	-,095	,120	,030	-,041	-,015	,012	,128	-,122
opnot	-,051	-,022	,046	,920	-,089	,075	-,040	,020	,000	,033	,065	-,021
cwhile	-,148	,095	,274	,710	,215	-,041	,032	,074	,037	-,110	-,143	,236

Figura 2. Cargas fatoriais

cif, cint, cmain, cscanf, celse), operadores aritméticos (+, -, *, ++, –), operadores lógicos (*opOr, opNOT*), operadores relacionais ($\geq$, $\leq$, $==$) e indicadores de compilação e execução (*compila, executa*) da Linguagem C.

Análise de Componentes Latentes

Fator	Características principais	Componente Latente
1	*opmais(+), opmult (*), opmenos (-)*	Operação Aritmética
2	*cbreak, ccase, cswitch*	Seleção
3	*opdec (–), cfor, opinc (++)*	Repetição Para
4	*opOR (||), opNOT (!), tchar*	Expressão Lógica
5	*cif*	Condição se verdadeira
6	*opmaiorigual ($\geq$)*	Comparação maior ou igual
7	*funciona, compila*	Execução
8	*cint, opmenorigual ($\leq$)*	Comparação menor ou igual
9	*cmain*	Inicío de programa
10	*cscanf*	Entrada de dados
11	*celse*	Condição se falsa
12	*opigual ($==$)*	Comparação igual

Tabela 2. Identificação de componentes latentes

Os fatores 1 a 12 da Tabela 2 foram nomeados conforme as maiores cargas fatoriais de cada grupo de componentes de habilidades. O Fator 1 (Figura 2), por exemplo, foi nomeado *Operação Aritmética* porque as componentes de maior carga fatorial que o formam são operadores aritméticos. Da mesma forma, o Fator 2 (Figura 2) foi nomeado *Seleção* porque as componentes *cbreak*, *ccase* e *cswitch* que possuem maior carga fatorial identificam, respectivamente, as palavras-chave *break, case* e *switch* da estrutura de seleção *switch* da Linguagem C.

Após a realização da análise fatorial, que reduziu o número de 35 componentes de habilidades de cada perfil a doze componentes latentes, a nova matriz reduzida e a matriz original foram submetidas ao algoritmo de *clustering Bisecting-kmeans*. Para a realização do *clustering*, foi utilizada a medida de similaridade *cosseno* e escolheu-se $k = 11$ para o número de *clusters* a serem formados.

A partir de cada *cluster* formado, foram selecionadas 2/3 de suas amostras para criar um modelo de regressão linear para estimar os *scores* dos 1/3 das amostras restantes de cada *cluster* [Oliveira 2013]. Conforme os valores de *scores* obtidos, os níveis de aprendizagem são classificados de acordo com a escala da Tabela 3.

Escala de Classificação

Scores	Classes
Acima de 4.0	A
De 2.5 a 4.0	B
De 1.5 a 2.5	C
Abaixo de 1.5	D

Tabela 3. Faixas de scores para classificação de níveis de aprendizagem

4.1. Resultados

A Tabela 4 apresenta os resultados de predição de *scores* para classificação de alunos através da análise dos erros médio, mínimo e máximo. Para os testes foram utilizadas a *Base-p* e o *Cluster-p*, que é o maior *cluster* formado da *Base-p*. A *Base-p* foi dividida em 52 amostras de treino (*Tr*) e 48 amostras de teste (*Te*). Já o *Cluster-p* foi dividido em quinze amostras de treino e sete amostras de teste. A Tabela 4 mostra os resultados sem (*sAF*) e com a análise fatorial (*cAF*), respectivamente.

Análise de erros

	Base-*p* (52Tr - 48Te)		Cluster-*p* (15Tr - 7Te)	
Erro	sAF	cAF	sAF	cAF
Erro Médio	0.7506	0.5964	1.4547	**0.5615**
Erro Mínimo	0.0096	0.0492	0.4300	**0.0383**
Erro Máximo	3.5000	3.5000	2.3000	**0.9731**

Tabela 4. Resultados de predição de scores

A Tabela 4 mostra que os *scores* de classificação são melhores preditos quando as estratégias de análise fatorial e de *clustering* são combinadas, conforme indicam os resultados do *Cluster-p* com análise fatorial. Por outro lado, a aplicação da análise fatorial não fez grande diferença nos resultados de predição de *scores* da *Base-p*.

A Tabela 5 apresenta os resultados de classificação de perfis de aprendizagem da *Base-p* e do *Cluster-p* sem e com a análise fatorial.

Acertos

	sAF	cAF
Base-*p*	57%	55%
Cluster-*p*	0	**85.7%**

Tabela 5. Resultados de classificação

Os resultados da Tabela 5 confirmam os resultados da Tabela 4, ao mostrar que as possibilidades de acertos das classes de níveis de aprendizagem são maiores combinando

as estratégias de análise fatorial e de *clustering*. Observa-se que, embora na *Base-p* os resultados de acertos de classes com e sem análise fatorial sejam semelhantes, no *Cluster-p*, sem análise fatorial não houve qualquer acerto na classificação das amostras de teste.

Os resultados iniciais de experimentação da combinação das técnicas de análise fatorial e de *clustering* em amostras reais de exercícios de programação apontam, portanto, para êxitos na classificação de perfis. Isso porque selecionam-se melhor as amostras de perfis (por *clustering*) e as componentes (por análise fatorial) para gerar os modelos de regressão utilizados na predição dos *scores* de classificação de níveis de aprendizagem.

5. Considerações Finais

Este trabalho apresentou uma estratégia de combinação das técnicas de análise fatorial e de *clustering* para mapeamento e classificação de perfis. Os primeiros resultados indicaram que essa estratégia pode ser aplicada para melhor selecionar amostras de perfis e componentes latentes de forma a melhorar os resultados de classificação de perfis.

A proposta dessa estratégia pode ser estendida a outros domínios, desde que se definam as componentes de habilidades que representem a aprendizagem de um domínio e que essas componentes possam ser quantificadas.

O mapeamento de perfis foi realizado identificando componentes latentes que explicam as relações entre as componentes de habilidades. Como trabalhos futuros a partir deste, sugere-se que se desenvolvam mais pesquisas em relação à representação de perfis de forma que esses realmente reflitam um modelo de aprendiz e que, reduzindo a dimensionalidade, de fato caracterizem a essência da aprendizagem de programação.

Espera-se, portanto, que os estudos realizados neste trabalho se consolidem em um passo inicial, mas relevante para a avaliação da aprendizagem de programação ao oferecer aos professores a possibilidade de compreenderem melhor a prática de programação de seus alunos e assim assisti-los individualmente.

Referências

Anderson, J. R. (2000). *Cognitive psychology and its implications*. Worth Publishers, New York and Basingstoke.

Anderson, T. W. (1984). *An Introduction to Multivariate Statistical Analysis*. Wiley, New York, NY, second edition.

Bishop, Y. M. M., Fienberg, S. E., and Holland, P. W. (1975). *Discrete Multivariate Analysis: Theory and Practice*. The MIT Press, Cambridge MA.

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*, LAK'11, New York, NY, USA. ACM.

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

Härdle, W. and Simar, L. (2012). *Applied Multivariate Statistical Analysis*. Springer.

Law, K. M. Y., Lee, V. C. S., and Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Comput. Educ.*, 55(1):218–228.

Looks, M., Levine, A., Covington, G., Loui, R., Lockwood, J., and Cho, Y. (2007). Streaming Hierarchical Clustering for Concept Mining. *Aerospace Conference, 2007 IEEE*, pages 1–12.

Mavrikis, M. (2010). *Machine–Learning Assessment of Students' Behavior within Interactive Learning Environments*, pages 441–449. Handbook of Educational Data Mining. CRC Press, Boca Raton, FL.

Mazza, R. and Dimitrova, V. (2007). CourseVis: A graphical student monitoring tool for supporting instructors in web–based distance courses. In *International Journal of Human–Computer Studies*, volume 65, pages 125–139, London, ROYAUME-UNI. Elsevier.

Mesic, V. and Muratovic, H. (2011). Identifying predictors of physics item difficulty: A linear regression approach. *Phys. Rev. ST Phys. Educ. Res.*, 7:010110.

Moreira, M. P. and Favero, E. L. (2009). Um ambiente para ensino de programação com feedback automático de exercícios. In *Workshop Sobre Educação em Computação (WEI)*, volume 17.

Morrison, D. F. (1990). *Multivariate Statistical Methods*. McGraw–Hill, third edition.

Naude, K. A., Greyling, J. H., and Vogts, D. (2010). Marking student programs using graph similarity. *Computers & Education*, 54(2):545 – 561.

Oliveira, M. and Oliveira, E. (2008). Avaliar para nivelar e formar: um sistema online de avaliação formativa para alunos de Biblioteconomia. In *Anais do XIX Simpósio Brasileiro de Informátiva na Educação (SBIE 2008)*, Fortaleza. SBC.

Oliveira, M. and Oliveira, E. (2014). Metodologia de Diagnóstico e Regulação de Componentes de Habilidades da Aprendizagem de Programação. In *CSBC 2014 - XXII WEI (Workshop sobre Educação em Computação)*, Brasília.

Oliveira, M. G. (2013). *Núcleos de Avaliações Diagnóstica e Formativa para Regulação da Aprendizagem de Programação*. Tese de doutorado, Universidade Federal do Espírito Santo.

Pea, R. D. and Kurland, D. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, 2(2):137 – 168.

Pillay, N. (2003). Developing intelligent programming tutors for novice programmers. *SIGCSE Bull.*, 35(2):78–82.

Pimentel, E., Franca, V., and Omar, N. (2003). A caminho de um ambiente de avaliação e acompanhamento contínuo de aprendizagem em programação de computadores. In *II Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG 2003)*, pages 212–213, Poços de Caldas, MG. Anais do II WEIMIG.

Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V., and Carlisle, M. (2006). Tools for teaching introductory programming: what works? *SIGCSE Bull.*, 38(1):560–561.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. KDD workshop on text mining.