Metodologia de Diagnóstico e Regulação de Componentes de Habilidades da Aprendizagem de Programação

Márcia G. de Oliveira, Elias Oliveira

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

clickmarcia@gmail.com, elias@acm.org

Abstract. *Computer programming is considered a difficult knowledge because it, to be learned, requires the combination of multiple skills and extensive practical of exercises. Furthermore, for a teacher, in large classes, it is practically impossible to assist each student individually and to give them feedbacks when a lot of exercises is applied. Knowing the formative assessment is a strategy that improves learning, we developed a programming assessment methodology which, supported by computational technologies, performs the following functions: to represent profiles of students by skills and to feed back these components by recommendation of activities until they reach acceptable performances of learning. This work contributes to programming learning providing a mechanism for diagnosis and adjustment of variables that represent the programming domain for better management of students'learning.*

Resumo. *A programação de computadores é considerada de difícil aprendizagem porque para ser aprendido requer a combinação de várias habilidades e extensa prática de exercícios. Para um professor, no entanto, é praticamente inviável acompanhar individualmente a aprendizagem de seus alunos bem como dar feedbacks para uma grande quantidade de exercícios em turmas numerosas. Sabendo que a avaliação formativa é uma estratégia que melhora a aprendizagem, desenvolvemos uma metodologia de avaliação para o domínio da programação de computadores que, apoiada por tecnologias computacionais, realiza as seguintes funções: representar perfis de alunos por componentes de habilidades e realimentar essas componentes por recomendação de atividades até que se alcancem desempenhos aceitáveis de aprendizagem. A contribuição deste trabalho para a aprendizagem de programação é oferecer um mecanismo de diagnóstico e regulação das variáveis que caracterizam a aprendizagem de programação para uma melhor gestão da aprendizagem de alunos.*

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. Esse processo envolve uma série de habilidades cognitivas como a compreensão do problema, a sequenciação lógica, a representação formal e a revisão o processo de resolução de um problema. Por isso, a programação de computadores é considerada um conhecimento de difícil aprendizagem.

Para [Anderson 2000], monitorando cuidadosamente as componentes de uma habilidade, é possível levar estudantes rapidamente ao domínio de habilidades mais complexas. Considerando as componentes de habilidades como um conjunto de variáveis que representam o domínio da aprendizagem de programação de computadores, o controle das componentes envolvidas no processo de programar pode ser realizado por modelos de avaliações diagnóstica e formativa [Perrenoud 1999]. A avaliação diagnóstica deve ter o papel de identificar habilidades e dificuldades de aprendizagem além de reconhecer perfis de alunos. A avaliação formativa, por sua vez, deve consistir de *feedbacks* e ajustes nos processos de ensino e de aprendizagem para alcançar objetivos traçados [Perrenoud 1999, Ballester 2003].

Neste trabalho desenvolvemos em tecnologias *web* e de reconhecimento de padrões uma metodologia para monitorar e regular as componentes de habilidades da aprendizagem de programação. Para realizar a avaliação diagnóstica, desenvolvemos o Núcleo de Avaliação Diagnóstica (NAD) que implementa a funcionalidade de mapeamento de perfis por componentes de habilidades medidas a partir de desempenhos em atividades. Para realizar a avaliação formativa, desenvolvemos o Núcleo de Avaliação Formativa (NAF), que implementa as funcionalidades de controle de estabilidade dos estados de aprendizagem e de recomendação de atividades de programação.

Em resumo, a proposta metodológica deste trabalho para monitoramento e controle da aprendizagem de programação consiste em representar perfis de alunos por componentes de habilidades e recomendar-lhes as atividades de programação conforme as dificuldades de aprendizagem identificadas em perfis.

Este trabalho está organizado conforme a ordem a seguir. Na Seção 2, apresentamos a fundamentação teórica da nossa proposta segundo uma abordagem sistêmica da avaliação. Na Seção 3, explicamos a nossa metodologia e as tecnologias que a apóiam. Na Seção 4, descrevemos como a metodologia foi aplicada em um turma real de programação. Na Seção 5, concluimos com as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

A avaliação, como instrumento adequado para regular e adaptar o ensino às necessidades e dificuldades de estudantes, deve cumprir três funções didático-pedagógicas: diagnóstica, formativa e somativa [Ballester 2003].

A avaliação diagnóstica deve ter o papel de identificar, através de um conjunto de variáveis representantes de perfis de alunos, habilidades e dificuldades de aprendizagem bem como classificar perfis conforme as medidas das variáveis de avaliação. A avaliação formativa, por sua vez, deve consistir de *feedbacks*, intervenções e ajustes no processo de ensino e de aprendizagem para alcançar objetivos traçados [Perrenoud 1999, Ballester 2003]. Já a avaliação somativa tem as funções de classificação e orientação de alunos, de reforçar êxitos e verificar resultados de um processo aprendizagem [Ballester 2003, Oliveira and Oliveira 2008].

Para o cognitivismo, a aprendizagem é contemplada não a partir de comportamentos observáveis, mas sim, do planejamento da ação e do tratamento das informações coletadas [Perraudeau-Delbriel 2009]. O processo de aprendizagem pode, sob esse ponto de vista, ser tratado como um sistema que recebe informações como entradas através do

ensino, gera saídas e é realimentado pelo planejamento da ação em função das informações coletadas nas saídas geradas.

Uma vez que a aprendizagem está no centro de uma rede de variáveis cujas interdependências agem sobre os efeitos [Perraudeau-Delbriel 2009], de acordo com a abordagem sistêmica, que é uma aplicação da Teoria Geral dos Sistemas (TGS) [L.BERTALANFY and Von 1973], a avaliação pode ser considerada um mecanismo realimentador dessa rede de variáveis. A Figura 1 ilustra o ensino, o processo de aprendizagem e a avaliação como partes de um sistema.

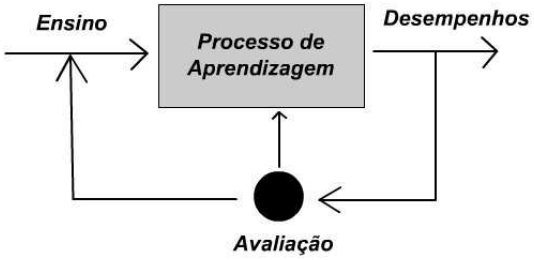

Figura 1. Avaliação sistêmica

Embora a utilização da abordagem sistêmica na prática avaliativa garanta êxitos de aprendizagem [Perrenoud 1999, Anderson 2000], na realidade, é praticamente inviável [Ballester 2003]. Mas as tecnologias chegaram como resposta ao questionamento de [Ballester 2003] sobre se é possível realizar avaliação formativa enfrentando as barreiras da grande quantidade de tarefas, do aumento do número de alunos, do aumento dos conteúdos estudados e da diversidade entre os estudantes. Somados as esses problemas, há os desafios do professor avaliar manualmente todos os exercícios dos estudantes e de dar-lhes *feedback* imediato de forma que aperfeiçoem suas soluções [Moreira and Favero 2009].

Nos últimos anos, várias tecnologias computacionais inteligentes têm sido desenvolvidas como apoio às metodologias de avaliações diagnóstica e formativa e têm promovido sucessos de aprendizagem [Anderson 2000, Mazza and Dimitrova 2007, De Oliveira et al. 2013].

No domínio da aprendizagem de programação, o *clustering*, a regressão linear e os algoritmos de aprendizagem supervisionada como o *kNN* e suas variações [Zhang and Zhou 2007, Manning et al. 2008] têm sido aplicados na correção automática de exercícios [Moreira and Favero 2009, Naude et al. 2010] bem como no mapeamento e na regulação da aprendizagem [De Oliveira et al. 2013] .

3. Núcleos de Avaliações Diagnóstica e Formativa da Programação

A metodologia de avaliação proposta neste trabalho têm como objetivo geral possibilitar melhor acompanhamento e remediação do processo de aprendizagem de programação para ajudar estudantes com dificuldades de aprendizagem a melhorarem seus desempenhos. Para isso, desenvolvemos um sistema de avaliação semi-automática para diagnóstico da aprendizagem de cada aluno e para recomendação de atividades conforme as dificuldades de aprendizagem reconhecidas em perfis de alunos.

A nossa metodologia de avaliação para o domínio da programação de computadores é apoiada por tecnologias computacionais de reconhecimento de padrões e reúne os

seguintes objetivos específicos:

1. Representar componentes de habilidades por desempenhos em atividades
2. Corrigir de forma semi-automática exercícios de programação
3. Mapear alunos em perfis de aprendizagem
4. Descobrir estados de aprendizagem dos alunos
5. Realimentar estados de aprendizagem dos alunos

Para alcançar esses objetivos, desenvolvemos uma estratégia de avaliações diagnóstica e formativa para monitorar e regular o processo de aprendizagem de programação.

A nossa estratégia reúne as funcionalidades de avaliações diagnóstica e formativa em um sistema de avaliação composto por dois núcleos: O Núcleo de Avaliação Diagnóstica (NAD) e o Núcleo de Avaliação Formativa (NAF), ambos implementados em tecnologias de reconhecimento de padrões como o *clustering* e o algoritmo de classificação multirrotulada ML-kNN [Zhang and Zhou 2007].

O NAD e o NAF comunicam-se com o mundo externo através do SOAP (Sistema *Online* de Atividades de Programação), que é um sistema *web* que possibilita ao professor disponibilizar tarefas para suas turmas e, ao aluno, realizar submissões de exercícios. Os exercícios submetidos, que são programas de computador em Linguagem C, são compilados e executados em um servidor remoto gerando as saídas para acesso do SOAP.

A Figura 2 representa a nossa proposta metodológica. Essa arquitetura mostra como o NAD e o NAF interagem com o SOAP para monitoramento e controle da aprendizagem de programação.

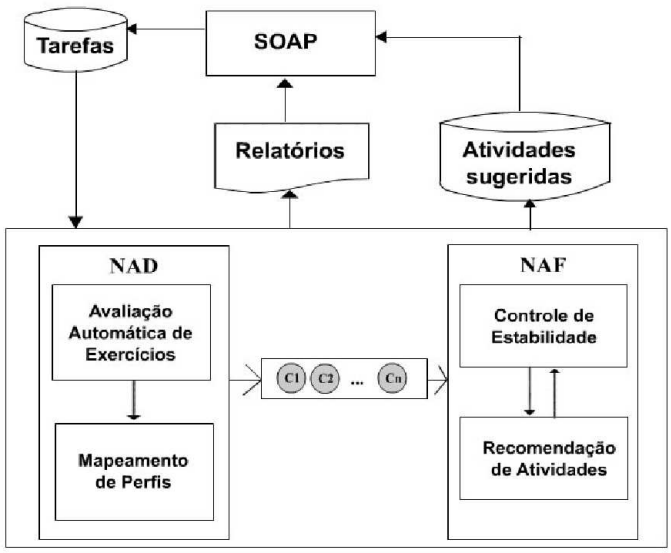

Figura 2. Arquitetura de avaliação semi-automática de programação

De acordo com a Figura 2, as atividades dos alunos, reunidas em tarefas, são recebidas e processadas pelos núcleos NAD e NAF. O NAD processa essas atividades gerando planilhas e mapas de perfis de aprendizagem que são entregues ao NAF na forma de componentes de habilidades C_i e, ao SOAP, na forma de relatórios. Esses relatórios

são visualizados por professores para que tenham melhor gestão do processo de aprendizagem de seus alunos. O NAF, com base no diagnóstico dos alunos fornecido pelo NAD, recomenda atividades para que os alunos com dificuldades possam melhorar seus desempenhos. As atividades sugeridas são apresentadas ao aluno através do SOAP.

Os nossos padrões de avaliações são as atividades de programação em Linguagem C desenvolvidas por alunos, cujos perfis são representados por seus desempenhos nessas atividades. Escolhemos a Linguagem C para aplicação desta metodologia porque é uma linguagem de programação adotada em muitos cursos de programação introdutória. No entanto, a metodologia desenvolvida neste trabalho pode ser estendida para outras linguagens de programação de paradigma de programação estruturado.

O NAD realiza as funções de correção semi-automática de exercícios de programação e de mapeamento de alunos em perfis de aprendizagem de acordo com as medidas de suas componentes de habilidades.

A correção semi-automática, que ainda está em desenvolvimento, é realizada a *priori* combinando métodos de *clustering* e modelos de regressão linear [Moreira and Favero 2009, Oliveira et al. 2010, Naude et al. 2010]. O *clustering* reúne em grupos os exercícios de programação com soluções semelhantes e indica que características explicam as semelhanças entre esses padrões. De cada *cluster*, selecionamos algumas amostras para o professor pontuar e através de um modelo de regressão linear atribuimos esse valor como nota das demais amostras reunidas em um mesmo *cluster*. Quando os *clusters* contêm apenas duas ou três amostras, seleciona-se para o professor atribuir nota apenas um padrão do *cluster* e as outras amostras do *cluster* são calculadas proporcionalmente ao índice de similaridade com a amostra pontuada pelo professor. Calculamos o índice de similaridade entre as amostras pela medida de similaridade *cosseno* [Manning et al. 2008].

O mapeamento de cada perfil de aluno é realizado a partir dos desempenhos dos alunos nas componentes de habilidades que representam o domínio de aprendizagem da programação. Esses desempenhos são obtidos por tarefas ou por atividades aplicadas em aula. Neste trabalho, a tarefa é um conjunto de atividades disponibilizadas em provas ou listas de exercícios. Cada atividade de programação requer como resposta do aluno um programa de computador desenvolvido em Linguagem C.

Para construir os gráficos de mapeamento de perfis utilizamos um algoritmo de *clustering* de abordagem hierárquica e com medidas de similaridade coeficiente de correlação (para tarefas) e cosseno (para atividades) [Steinbach et al. 2000, Manning et al. 2008]. Na próxima seção, apresentamos exemplos desses mapas de perfis.

O NAF possui as funções de controle de estabilidade (ou nivelamento) de aprendizagens e de recomendação de atividades para alunos que apresentam dificuldades de aprendizagem apontadas pelas componentes de habilidades. A função de controle de estabilidade apenas verifica se os desempenhos obtidos por um aluno nas componentes de habilidades satisfazem um estado-objetivo de aprendizagem. Se sim, o processo de recomendação de atividades é encerrado.

A recomendação de atividades do NAF é realizada reformulando o problema de recomendação em um problema de classificação multirrotulada em que cada perfil de aluno é associado a uma ou mais classes de atividades de programação [De Oliveira et al. 2013].

Para isso, o sistema analisa perfis multidimensionais de alunos e, com base em um histórico de recomendações feitas manualmente, recomenda-lhes classes de atividades conforme as similaridades com perfis que já receberam recomendações.

Para identificar e recomendar classes de atividades de acordo com as variáveis de avaliação que precisam ser melhoradas, o classificador ML-kNN foi aplicado ao *ds-FAR*, um *dataset* com 1784 programas em Linguagem C submetidos por alunos. Esses programas foram mapeados em 37 componentes de habilidades e associadas a 19 classes de atividades de programação. A Figura 3 é um exemplo de como três perfis de alunos foram mapeados nessas variáveis e como as classes de atividades foram recomendadas.

<table>
<tr><th></th><th colspan="37">assessment variables (attributes)</th></tr>
<tr><th>Profiles</th><th>compile</th><th>run</th><th>printf</th><th>include</th><th>main</th><th>return</th><th>+</th><th>-</th><th>*</th><th>/</th><th>%</th><th>++</th><th>--</th><th>=</th><th>float</th><th>int</th><th>char</th><th>scanf</th><th>></th><th><</th><th>!=</th><th>==</th><th><=</th><th>>=</th><th>||</th><th>!</th><th>&&</th><th>if</th><th>else</th><th>else if</th><th>switch</th><th>case</th><th>break</th><th>default</th><th>do</th><th>while</th><th>for</th></tr>
<tr><td>A1</td><td>1</td><td>1</td><td>0.3</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0.1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0.5</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0.5</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>A2</td><td>1</td><td>1</td><td>2.4</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0.2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1.5</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>5</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2.5</td><td>2.5</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>A3</td><td>1</td><td>1</td><td>0.3</td><td>3</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td><td>3</td><td>3</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0.5</td><td>0.5</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

(A)

	classes of activities																		
Profiles	nothing	structure	output	arithmetic operators	types	assignment	input	comparison operators	logic operators	if structure	if-ladder	switch structure	while structure	do-while structure	for structure	understanding	depuration	execution	efficiency
A1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
A3	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0

(B)

Figura 3. (A) Mapeamentos de perfis em variáveis de avaliação (B) Classes de Atividades (1 = Recomendada; 0 = Não-recomendada)

Na Figura 3, os três estados de perfis $A1$, $A2$ e $A3$ são representados pelos desempenhos dos alunos em uma atividade de programação.

Em síntese, a nossa estratégia combina técnicas de reconhecimento de padrões para classificar alunos e recomendar-lhes atividades conforme os seus perfis. O monitoramento e o controle das componentes de habilidades da aprendizagem de programação através dessa estratégia possibilita ao professor realizar melhor gestão da aprendizagem individual de seus alunos.

4. Estudo de Caso

Alguns módulos da nossa estratégia de avaliação semi-automática como o mapeamento de perfis e a recomendação de atividades foram experimentados em turmas de programação da Universidade Federal do Espírito Santo (UFES) a partir de programas em Linguagem C desenvolvidos por alunos ao longo de um semestre.

Através dos mapas de perfis gerados, o professor pôde identificar classes de alunos que apresentam as mesmas características em relação às componentes de habilidades e reconhecer dificuldades e habilidades dos alunos de uma turma. São exemplos desses mapas as Figuras 4 e 5.

Além do mapeamento de perfis de alunos, aplicando a mesma estratégia de *clustering*, podemos realizar diagnósticos de questões, de tarefas e de provas para avaliar o nível de dificuldade das questões bem como apontar os alunos hábeis para resolvê-las.

Os gráficos de *recall* (*R*) e *precision* (*P*) da Figura 4 são exemplos de mapeamento de perfis baseado em tarefas. Nesse tipo de mapeamento, as componentes de habilidades são representadas por desempenhos medidos em *recall* e *precision* em listas ou provas que sintetizam unidades dos conteúdos de programação estudados. O *recall* é uma medida que expressa a razão entre o número de acertos n_a e o total de exercícios n_t de uma tarefa. Já a medida *precision* informa a razão entre o número de acertos n_a e número n_r de exercícios resolvidos por um aluno. Os cálculos dessas medidas são obtidos conforme as equações a seguir [Manning et al. 2008]:

$$R = \frac{n_a}{n_t}, P = \frac{n_a}{n_r}, \text{ para } (n_r \leq n_t)$$

Nos gráficos da Figura 4, as linhas representam os alunos e as colunas, cada tarefa aplicada ao longo de um curso de programação. As áreas em vermelho indicam desempenhos acima de 70%. Já as áreas verdes indicam ausência de desempenhos ou desempenhos inferiores a 70%.

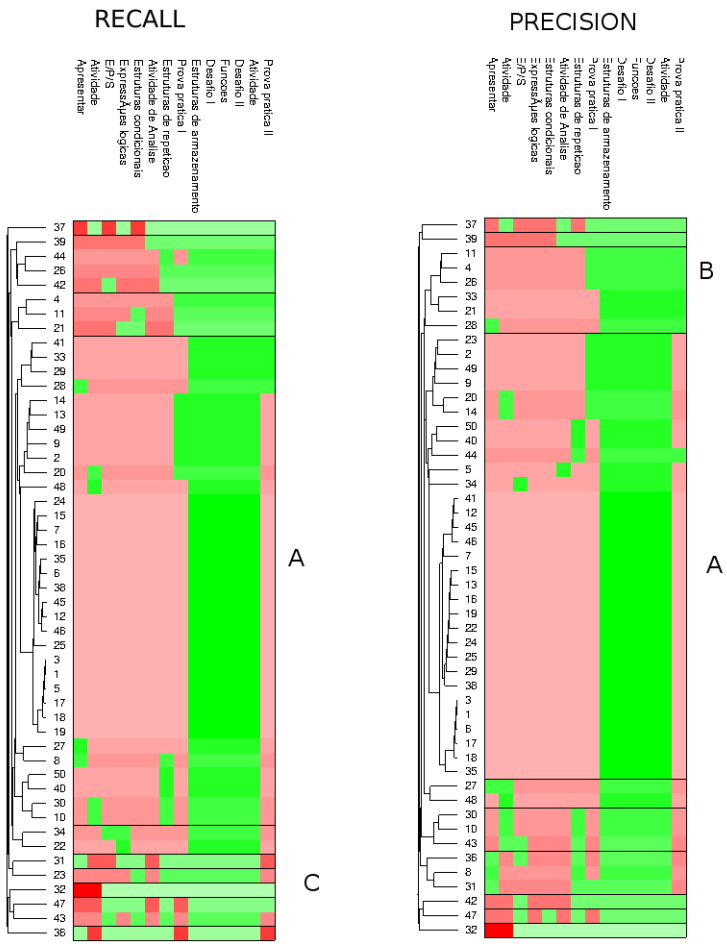

Figura 4. Perfil de aluno baseado em tarefas

Através dos gráficos da Figura 4, reconhecem-se diferentes classes de alunos, entre as quais citamos: as classes dos melhores alunos, alunos que se fazem as tarefas esporadicamente e os alunos com dificuldades em fazer provas.

Nos gráficos da Figura 4, os alunos que apresentam os melhores desempenhos nas tarefas são identificados pela letra A. Esses alunos fazem todas as tarefas com desempenhos acima de 70% e reproduzem os bons resultados nas provas práticas, que são as últimas colunas das faixas em vermelho, isto é, as colunas em vermelho antes do início da faixa verde (tarefas não avaliadas) e depois do seu final.

No gráfico de *recall* da Figura 4, nas linhas da parte inferior estão os alunos identificados por C, que são aqueles alunos que fazem as tarefas esporadicamente. No gráfico de *precision*, nas linhas superiores, estão os alunos identificados pela letra B, que são os alunos que fazem bem todas as tarefas, mas não alcançam desempenhos satisfatórios nas provas práticas.

A Figura 5 é um mapa de diagnóstico dos alunos de uma turma em uma atividade de programação. Nesse gráfico, as linhas representam os alunos de uma turma, que são identificados por um número. As colunas são as componentes de habilidades representadas por desempenhos no uso de indicadores de execução (*compile, run*), palavras e funções reservadas da Linguagem C (*include, main, int, float, scanf, return, float, return, if, for, while, switch*) e operadores de atribuição (=), aritméticos (+, -, *,/, %), de comparação (==, >,<,>=,<=) e lógicos (&&, ||, !). As áreas em vermelho apontam as dificuldades de aprendizagem nas componentes de habilidades.

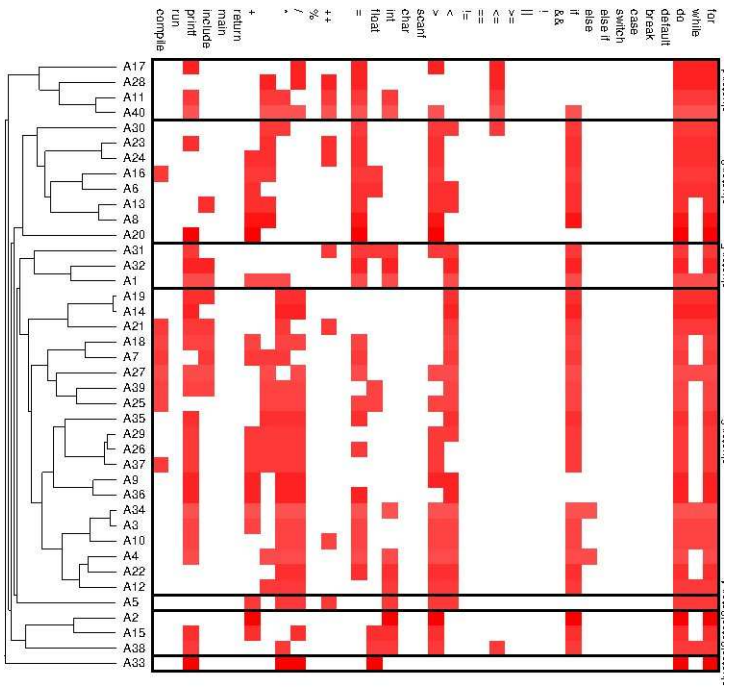

Figura 5. Perfil de aluno baseado em atividades

Para indicar as dificuldades de aprendizagem nas áreas vermelhas do gráfico da Figura 5, foi estabelecida uma solução como gabarito. Cada componente de habilidade foi medida calculando-se a razão entre a sua frequência de ocorrência em soluções desen-

volvidas por alunos e a sua frequência de ocorrência na solução-gabarito. Se essa razão resulta em um valor abaixo de 0.7, isto é de 70%, a componente de habilidade é sinalizada de vermelho indicando dificuldade de aprendizagem. Da mesma forma, se a medida da componente de habilidade exceder a 1, isto é, a 100%, a componente também é sinalizada de vermelho entendendo-se que o aluno usou instruções de programação acima do limite esperado, o que caracteriza ineficiência no uso da linguagem de programação.

Através da estratégia de mapeamento de perfis da Figura 5, geramos 1784 estados de perfis de alunos a partir de programas desenvolvidos em Linguagem C. Em seguida, experimentamos o sistema de recomendação do NAF recomendando atividades conforme classes de atividades já recomendadas a outros perfis similares.

A nossa estratégia de recomendação alcançou 96% de acurácia e 89% de precisão média. O *dataset ds-FAR* com os perfis de estudantes e as recomendações, os procedimentos e os resultados de performance do algoritmo Ml-kNN aplicado nessa estratégia podem ser visualizados detalhadamente em [De Oliveira et al. 2013].

5. Considerações finais

Este trabalho apresentou uma estratégia computacional de avaliações diagnóstica e formativa para regulação da aprendizagem de programação através do mapeamento de perfis e da recomendação de atividades.

As principais contribuições do modelo de avaliação semi-automática de programação deste trabalho são as seguintes: combinamos técnicas de reconhecimento de padrões para avaliação de alunos em conformidade com padrões de professores; formulamos o problema de recomendação de atividades em um problema de classificação multirrotulada; criamos uma representação de perfis por tarefas por medidas de *recall* e *precision* para identificar classes de alunos e suas dificuldades; criamos outra representação de perfis por atividades que caracteriza a aprendizagem de programação a partir da frequência de uso dos recursos da Linguagem C; formamos uma base de 1784 exercícios de programação resolvidos por alunos e avaliados por professores para realização de experimentos de avaliação e de recomendação semi-automática de atividades.

A proposta de nossa metodologia 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.

Como trabalhos futuros a partir deste trabalho, sugerimos que se desenvolvam mais pesquisas em relação à representação de perfis de forma que realmente reflitam um modelo do aprendiz e os seus estados de aprendizagem. Esse modelo ajudaria o sistema de recomendação de atividades a traçar uma rota de aprendizagem através da prática de exercícios conforme as condições de aprendizagem do aluno.

Através das contribuições e dos trabalhos futuros visamos reduzir esforços de professor na correção de exercícios e no acompanhamento individual de seus alunos. Para os alunos, oferecemos possibilidades de *feedbacks* mais rápidos e uma verdadeira prática assistida da programação.

Referências

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

Ballester, M. (2003). *Avaliação como apoio à aprendizagem*. Artmed, Porto Alegre, RS. Trad. Valério Campos.

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*.

L.BERTALANFY and Von, L. (1973). *Teoria geral dos sistemas*. Petrópolis: Vozes.

Manning, C., Raghavan, P., and Schutze, H. (2008). *Introduction to information retrieval*. Editora Thomson, Cambridge University Press.

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.

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.

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

Oliveira, E., Fraga, N., Oliveira, M., and Marchesi, R. (2010). Uma tecnologia de agrupamento de respostas para redução de esforço de correção de atividades em sistema online de apoio à avaliação formativa em indexação. In *XI ENANCIB:Encontro Nacional de Pesquisa em Ciência da Informação*, Rio de Janeiro. ENANCIB.

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.

Perraudeau-Delbriel, M. (2009). *Estratégias de aprendizagem: como acompanhar os alunos na aquisição dos saberes*. ARTMED.

Perrenoud, P. (1999). *Avaliação: da excelência à regulação das aprendizagens – Entre Duas Lógicas*. Artmed Editora, Porto Alegre, RS.

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

Zhang, M.-L. and Zhou, Z.-H. (2007). ML-KNN: A Lazy Learning Approach to Multi-label Learning. *Pattern Recognition*, 40(7):2038 – 2048.