Ensino e Aprendizagem de Algoritmos com o AlgoLC

Patrícia Gerent Petry[1], Marta Costa Rosatelli[2]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
INE-CTC, Cx.P. 476, Campus Universitário – 88040-900 – Florianópolis – SC – Brasil

[2]Programa de Pós-Graduação em Informática – Universidade Católica de Santos
R. Dr. Carvalho de Mendonça, 144 – 11070-906 – Santos – SP – Brasil
patricia@mbox1.ufsc.br, rosatelli@unisantos.br

Abstract. *This paper presents AlgoLC, a system that supports teaching and learning algorithms. AlgoLC provides the teacher with elements to better identify the students' doubts and errors when they are writing an algorithm. The learning companion interacts with the student helping to identify and correct his or her own mistakes. AlgoLC is characterized as a Learning Companion System.*

Resumo. *Este artigo apresenta o AlgoLC, um sistema que dá suporte ao ensino e aprendizagem de algoritmos. O AlgoLC fornece para o professor elementos que permitem identificar as dúvidas e erros dos estudantes quando escrevem um algoritmo. O companheiro de aprendizagem interage com o estudante auxiliando-o a identificar e corrigir seus próprios erros. O AlgoLC é caracterizado como um Sistema Companheiro de Aprendizagem.*

1. Introdução

A disciplina de algoritmos, que faz parte do currículo de alguns cursos de graduação da área tecnológica, é uma barreira para muitos estudantes que se iniciam na área. Em um algoritmo, o estudante precisa desenvolver seu raciocínio lógico propondo soluções de problemas.

Este artigo apresenta o AlgoLC, um Sistema Companheiro de Aprendizagem para auxílio ao ensino e aprendizagem de algoritmos. O artigo está organizado como se segue. A seção 2 descreve trabalhos relacionados. A seção 3 apresenta uma visão geral dos Sistemas Companheiro de Aprendizagem. A seção 4 introduz a Modelagem Baseada em Restrições. A seção 5 apresenta o AlgoLC. A seção 6 relata uma avaliação feita com o protótipo do AlgoLC. Finalmente, a seção 7 apresenta conclusões e direções para trabalhos futuros.

2. Trabalhos Relacionados

Um dos grandes problemas do modo tradicional de ensino em algoritmos é a dificuldade de motivar os estudantes, de fazer com que se interessem pela disciplina e entendam que seu conteúdo é importante e fundamental como base para outras disciplinas. No contexto

da aprendizagem baseada em computador, alguns sistemas foram desenvolvidos para auxiliar o ensino e aprendizagem de algoritmos.

O HabiPro (*Habits of Programming*) (Vizcaíno *et al*., 2000) é ambiente colaborativo que visa desenvolver nos estudantes "bons hábitos" em programação. Ele não ensina a programação, mas estimula os estudantes novatos em programação a adquirirem habilidades como a observação e reflexão sobre a estrutura do algoritmo, necessários para se tornarem bons programadores. A interface desta aplicação tem duas janelas: um *chat* para a comunicação entres os estudantes e uma área de trabalho onde os estudantes devem colaborar para resolver um determinado problema de programação. Após a proposta da solução do grupo ser apresentada, se esta não está correta, o sistema propõe quatro tipos de ajuda: oferece suporte ao estudante sobre como resolver o problema; mostra a solução e uma explicação do porque o problema ter sido resolvido com aquela técnica; mostra um exemplo similar do problema que o estudante tentou resolver e sua solução; ou mostra a solução do problema.

Um outro protótipo de sistema de ensino e aprendizagem de programação implementado em Delphi, proposto por Kemp *et al*. (2003), apresenta programas de computadores previamente prontos e o estudante deve criar um diagrama corresponde ao programa já implementado. Neste sistema, os estudantes devem associar as estruturas de programação (se-então-senão, para-faça, enquanto-faça) com a representação correspondente aos diagramas. O sistema apresenta dois níveis de dificuldade para o estudante. No primeiro nível, o estudante seleciona um pedaço do código previamente escrito e move-o para ser trabalhado como um diagrama. O sistema faz a conversão de forma automática para o inglês e o coloca numa caixa com o formato apropriado. O estudante deve colocar a caixa na posição correta e fazer os *links* com os outros elementos do diagrama. No nível mais difícil, o próprio estudante tem que selecionar a forma da caixa.

No que diz respeito aos SCAs, verifica-se que foram desenvolvidos sistemas em vários domínios. Entretanto, não existe um consenso absoluto na literatura sobre a composição das técnicas e tecnologias utilizadas no projeto de um SCA. A complexidade dos sistemas faz com que cada implementação seja feita de acordo com o objetivo e respectiva(s) funcionalidade(s) requerida (s). Sendo assim, em cada sistema ou protótipo são definidos a composição do SCA, o comportamento dos CAs, o mecanismo de comunicação, e a modelagem do estudante, do tutor e do domínio (Faraco *et al*., 2004b).

De acordo com Rasseneur *et al*. (2002) o AMICO (*Apprentissage des Mathématiques par Interaction avec des Compagnions*), é um protótipo de um sistema para aprimorar o aprendizado da matemática através da interação entre um estudante e vários companheiros virtuais. Ele foi projetado para ser utilizado na sala de aula durante sessões individuais de ajuda ou quando os estudantes trabalham em pares. O objetivo pedagógico é obter vários tipos de justificativas e usar diferentes modos de representação nos exercícios. O AMICO foi implementado em Java utilizando a UML (*Unified Modeling Language*) para desenvolver sua arquitetura. O software oferece dois cenários diferentes de interação com companheiros virtuais: competitivo e cooperação. Foram implementadas quatro características para os CAs. Estas características seguem adaptações para diferentes estilos de aprendizagem e estratégias de ensino.

O projeto de desenvolvimento de LuCy surgiu da necessidade da presença de características dos STIs e de sistemas de aprendizagem colaborativa em um único ambiente. LuCy foi desenvolvido para um STI já existente, denominado PROPA, cujo domínio de conhecimento era o ensino de habilidades em análise exploratória das atividades de satélites. Os analistas de atividades de satélites usam a análise exploratória para interpretar atividades de satélites baseados em informações incompletas, incertas ou inconsistentes. As habilidades necessárias incluem a formulação de hipóteses, a busca por evidências e a avaliação da veracidade e significado dessas evidências. O CA LuCy é um par interativo que responde questões e oferece sugestões da mesma forma que um estudante colaborador real o faria. A vantagem de LuCy, se comparada com um par real, é sua total disponibilidade para com o estudante humano e sua capacidade de apoiá-lo em dúvidas particulares. A funcionalidade de LuCy está baseada em uma arquitetura onde se fazem presentes um *parser* para tratar as sentenças dos estudantes, o modelo do domínio do conhecimento, o modelo do estudante com histórico de diálogos, um selecionador e um gerador de respostas (Goodman *et al*., 1998).

O LeCo-EAD (Faraco *et al*., 2004a, 2004b) é um SCA para o ensino à distância baseado na *Web*, que é composto por múltiplos CAs. Estes têm com comportamentos próprios e estão permanentemente disponíveis para interagir com os estudantes. O ambiente virtual de aprendizagem contempla diferentes estratégias de ensino, representadas pelos CAs do tipo colaborador, aprendiz e *trouble maker*. A partir do perfil do estudante identificado por uma escala de atitudes, o LeCo-EAD sugere qual tipo de CA irá atuar como seu companheiro virtual de aprendizagem. O LeCo-EAD é um SCA que adota dois tipos de adaptação: a de conteúdo e a de estratégia de ensino. A adaptação de conteúdo é realizada a partir da performance do estudante no curso, em que o sistema, através de um esquema de pré-requisitos, apresenta somente os conceitos que o estudante está apto a desenvolver. O outro tipo de adaptação ocorre quando o sistema solicita a participação do estudante na escolha do CA mais adequado ao seu perfil, inicialmente através da escala de atitudes e durante o curso através dos mecanismos de *feedback*.

Por fim, numa perspectiva diferente, Kim (2005) investiga o potencial dos agentes pedagógicos como CAs para construir relações sociais com os estudantes e conseqüentemente motivar a aprendizagem

3. Sistemas Companheiro de Aprendizagem

Os Sistemas Companheiro de Aprendizagem (SCAs) são Sistemas Tutores Inteligentes (STI), que prevêem, além dos módulos tradicionais da arquitetura de um STI, um Companheiro de Aprendizagem (CA) (Chan & Baskin, 1990). Os CAs são pares virtuais que apóiam os estudantes durante o processo de ensino-aprendizagem.

Os SCAs são baseados na teoria sócio-cultural pedagógica proposta por Vygotsky (1978), em que a construção do conhecimento implica em uma ação compartilhada. Através de interações sociais com o meio, ou com outros pares semelhantes e, eventualmente mais capazes, a relação entre sujeito e objeto de conhecimento é estabelecida, tornando o indivíduo capaz de realizar atividades que, sem a ajuda externa, representariam uma dificuldade muito grande. Os diferentes ritmos, comportamentos, valores, experiências e níveis de conhecimento permitem a cooperação e também a

competição nas interações, que são elementos importantes para a melhoria das capacidades individuais.

O modelo do tutor em um SCA tem como função coordenar as atividades de aprendizagem, a entrega do material didático, a escolha do CA adequado para cada estudante, a distribuição de responsabilidades entre os estudantes. Ao contrário dos STIs, a interação com os estudantes é feita através do CA.

Os SCAs fazem uso da abordagem pedagógica proposta por Vygotsky, segundo a qual através das interações sociais com o meio ou com outros pares semelhantes e eventualmente mais capazes, a relação entre sujeito e conhecimento é estabelecida (Faraco *et al.*, 2004a). O papel do CA é ser um companheiro virtual para o estudante, interagindo com ele a fim de ajudar no seu aprendizado, da mesma forma que outro companheiro humano o faria. Uma vantagem que é importante ser destacada é a total disponibilidade do CA para com os estudantes, incentivando-os a interagirem com o sistema e aprenderem colaborativamente. O papel tradicional de tutoria (como acontece em um STI por exemplo), no qual o tutor "toma conta" do estudante durante o seu progresso através do conteúdo, é desempenhado, no caso de um SCA, por um CA do tipo colaborativo (Goodman *et al.*, 1998).

Um CA pode assumir diferentes papéis e comportamentos ao interagir com os estudantes. Um CA colaborativo atua junto ao estudante na resolução de problemas (Chan & Baskin, 1990; Regnemalm, 1996) enquanto o CA competitivo trabalha na solução dos problemas de modo independente e simultâneo ao estudante. O CA pode se comportar ainda como um aprendiz do estudante humano. Conseqüentemente, este estudante necessita revisar, refletir e esclarecer seu próprio conhecimento para que consiga transmiti-lo ao seu par virtual (Uresti, 2000). O CA que é um causador de problemas (*trouble maker*) tenta motivar o estudante através da provocação, a fim de indicar suas deficiências na matéria sendo ensinada (Hietala & Niemirepo, 1998).

O CA, no AlgoLC, colabora com o estudante no processo de aprendizagem, auxiliando-o no desenvolvimento dos algoritmos através da observação de seus erros. O aprendizado é facilitado porque o CA ajuda o estudante a identificar e corrigir seus próprios erros. O AlgoLC usa a Modelagem Baseada em Restrições para dar suporte ao seu raciocínio e intervenções, que são mensagens de *feedback* apresentadas ao estudante.

4. Modelagem Baseada em Restrições

A abordagem da Modelagem Baseada em Restrições (MBR) (Ohlsson, 1994) tem como fundamento uma teoria de aprendizagem que se baseia na identificação dos erros. Esta teoria é relevante para os estudantes que não detêm o conhecimento de um determinado assunto e, portanto, não são capazes de detectar os próprios erros. A MBR representa o conhecimento sobre um domínio através de um conjunto de restrições de estados para a solução correta dos problemas deste domínio (Mitrovic & Ohlsson, 1999). As restrições são divididas em todas as possibilidades de soluções corretas e incorretas para um problema. Cada restrição representa um conceito declarativo que deve ser aprendido e analisado pelo estudante (Martin & Mitrovic, 2003). Isto significa que o domínio é representado através dos efeitos (estados da resolução do problema) que as ações do estudante podem gerar, a partir da suposição de que é possível identificar um estado do problema que viola conceitos fundamentais do domínio.

As restrições propostas por Ohlsson (1994) são constituídas por um par ordenado (*Cr*, *Cs*), no qual *Cr* é a condição de relevância e *Cs* a condição de satisfação. A condição de relevância é usada para identificar um conjunto de estados de problema para o qual determinada restrição se aplica e para o qual um possível erro pode ocorrer (ou seja, a restrição pode ser violada). A condição de satisfação é usada para identificar um subconjunto desses estados relevantes nas quais a restrição é satisfeita, ou seja, a condição que indica o acerto do estudante com relação à restrição. Então, se *Cr* é satisfeita em um estado do problema, para que esse estado esteja correto é preciso que *Cs* também seja satisfeita (ou então alguma coisa está errada). Por exemplo, no caso de se fazer a leitura de uma variável num algoritmo, a condição de relevância e a condição de satisfação podem ser:

> *Cr*: Se o objetivo é ler uma variável então
>
> *Cs*: esta variável deve estar declarada como um inteiro, real, *string* ou lógica (senão ocorre um erro).

Se o estudante entra com uma solução, e *Cr* for satisfeita, mas *Cs* não, dizemos que a restrição (*Cr*, *Cs*) foi violada, indicando assim que o estudante cometeu um erro. Segundo Martin & Mitrovic (2002), na MBR não estamos interessados no que o estudante fez ao longo do processo de solução de um problema, mas em que estado ele se encontra atualmente. Portanto, o modelo do domínio é uma coleção de estados de descrições na forma de:

> SE <condição de relevância> é verdade para a solução do estudante,
>
> ENTÃO <condição de satisfação> deve ser também verdade, senão a solução está errada.

As restrições que modelam o conhecimento do domínio servem para classificar estados de problemas. Ou seja, identificam que o estudante deve receber uma determinada ação de instrução através do sistema, sugerindo novas idéias que estejam relacionadas àquela determinada restrição (Martin & Mitrovic, 2002). Essa abordagem pode ser usada tanto para modelar o conhecimento do domínio como para modelar o conhecimento do estudante neste domínio, através das restrições violadas.

No caso do sistema AlgoLC o estudante tem um auxílio individualizado de um CA que envia mensagens de *feedback*, estimulando o estudante a verificar seus erros e corrigi-los, ao contrário de, por exemplo, propor a melhor solução.

5. Descrição do AlgoLC

5.1 Arquitetura

A Figura 1 apresenta a arquitetura do AlgoLC (Petry & Rosatelli, 2006). O domínio é representado com restrições. Ou seja, o modelo de domínio contém as informações corretas que o estudante deverá aprender e é a base de conhecimento do sistema. O AlgoLC é capaz de fazer análises sobre o que o estudante sabe e como ele está progredindo na construção de um algoritmo.

O modelo do estudante inclui as características gerais do estudante (nome, histórico, etc.) e também o conhecimento correto ou incorreto do estudante: quais

restrições o estudante violou ou não. Esse registro é utilizado pelo modelo do tutor para determinar algumas estatísticas importantes, como por exemplo, onde o estudante mais errou e qual restrição foi mais violada.

Os problemas apresentados são escolhidos pelo AlgoLC com base no modelo do estudante. Nesse modelo o conhecimento do estudante é representado como uma sobreposição (*overlay*) do modelo de domínio. Para cada problema a ser resolvido no sistema, um modelo individual de sobreposição armazena o nível de conhecimento do estudante sobre aquele problema. Esse nível de conhecimento pode representar se o estudante sabe ou não sabe, uma medida qualitativa do que ele sabe, ou uma medida quantitativa do quanto o estudante conhece o assunto. Cada estudante possui associado seu modelo com informações referentes ao seu desempenho no desenvolvimento dos problemas apresentados pelo sistema. Os critérios utilizados para armazenar informações referentes aos resultados obtidos no desenvolvimento de problemas pelo estudante são: o número de vezes em que o CA interfere no desenvolvimento dos problemas; o número de acertos no desenvolvimento dos problemas propostos; o número vezes em que ocorreram violações das restrições na resolução dos problemas e quais as restrições que foram mais violadas pelo estudante.

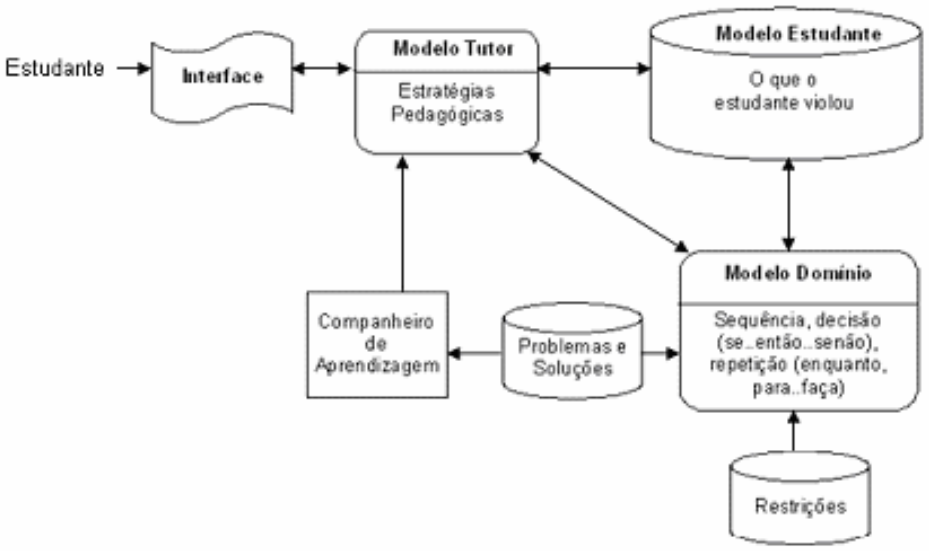

Figura 1. Arquitetura do AlgoLC

O modelo do tutor possui as estratégias pedagógicas que tratam do conhecimento sobre o ensino daquele domínio.

A interface com o usuário é o meio pelo qual o estudante se comunica com o sistema. Esta apresenta o material didático ao estudante e captura informações do mesmo, a fim de que o sistema possa monitorar o seu progresso e o seu comportamento e mantenha o modelo do estudante atualizado.

Quando o estudante submete uma solução (i.e., um algoritmo), esta é passada ao modelo do estudante. O modelo do estudante determina, primeiramente, quais restrições são relevantes para a solução atual e, em seguida, quais restrições são satisfeitas. Ou seja, o que foi violado e o que não foi. O CA intervém automaticamente, através de

mensagens ao estudante, de acordo com as restrições que foram violadas. O CA conduz o estudante aos desafios propostos na resolução dos problemas, observando e verificando todas as ações geradas pelo estudante na interface e intervindo quando uma restrição foi violada. O CA no AlgoLC é autônomo e colaborativo, sempre interagindo com o estudante quando for apropriado, sem ser solicitado pelo estudante. O modelo do tutor interage com o CA, escolhendo as mensagens de *feedback* nos diferentes casos através da violação das restrições.

5.2 Restrições Modeladas

O modelo de domínio do AlgoLC consiste de um conjunto e 33 (trinta e três) restrições referentes a 6 (seis) tipos de problemas identificados como usuais e freqüentes no ensino de algoritmos. Outras restrições podem ser adicionadas ao protótipo, a partir desse conjunto inicial.

As restrições modeladas foram agrupadas em quatro níveis que dizem respeito ao grau de dificuldade de um exercício. O estudante tem a liberdade de escolher por qual nível iniciar seus exercícios:

1: declaração de variáveis: algoritmos que envolvem somente a declaração de variáveis.

2: estruturas seqüenciais: algoritmos que envolvem tanto declaração de variáveis, quanto estruturas seqüenciais.

3: estruturas de decisão: algoritmos que envolvem declaração de variáveis, estruturas seqüenciais e estruturas de decisão.

4: estruturas de repetição: algoritmos que envolvem declaração de variáveis, estruturas seqüenciais, estruturas de decisão e estruturas de repetição.

No início do trabalho com o AlgoLC, o estudante escolhe um grupo de problemas. A Figura 2 mostra a interface com o usuário do AlgoLC. A interface é dividida em duas áreas: exercícios e algoritmo. Na área de exercícios o estudante faz a escolha do exercício desejado. Na área denominada algoritmo o estudante digita a solução do problema proposto. O estudante tem a opção de listar todos os exercícios clicando no menu usuário e em seguida no item exercícios, ou escolher o nível do exercício que ele deseja resolver. Após a escolha do nível, o estudante pode optar pelo o tipo de exercício que deseja resolver. A solução deve ser digitada na área à direita. Após a digitação de cada linha do algoritmo como solução proposta (utilizando a tecla <enter> para pular de linha), o CA é acionado automaticamente caso alguma restrição naquela linha tenha sido violada. Uma mensagem de *feedback* associada à restrição violada é enviada ao estudante.

Na barra de menu têm-se as seguintes opções: usuário, companheiro e sair. Na opção *usuário* o estudante pode trocar de usuário ou visualizar todos os exercícios. Na opção *companheiro*, têm-se as opções “analisar algoritmo” e a opção “relatórios”. A opção *analisar algoritmo* faz a verificação linha a linha da solução proposta pelo estudante. A opção *relatórios* gera um relatório do estudante, que permite verificar quais restrições e quantas vezes o estudante a violou durante o exercício, e também um relatório geral, que mostra a quantidade de vezes que cada restrição foi violada por aquele estudante na solução dos exercícios já resolvidos.

6. Avaliação

O protótipo do AlgoLC foi testado com estudantes de um curso de graduação em Ciência da Computação: 15 estudantes que cursam o primeiro ano da disciplina de algoritmo. Dos quinze estudantes que realizaram o teste, 9 (nove) eram repetentes desta disciplina. Os estudantes não tinham experiência prévia com a disciplina de algoritmo, visto que apenas um dos estudantes já tinha realizado um curso de lógica de programação. Os outros estudantes tiveram o primeiro contato com este domínio apenas nesta disciplina.

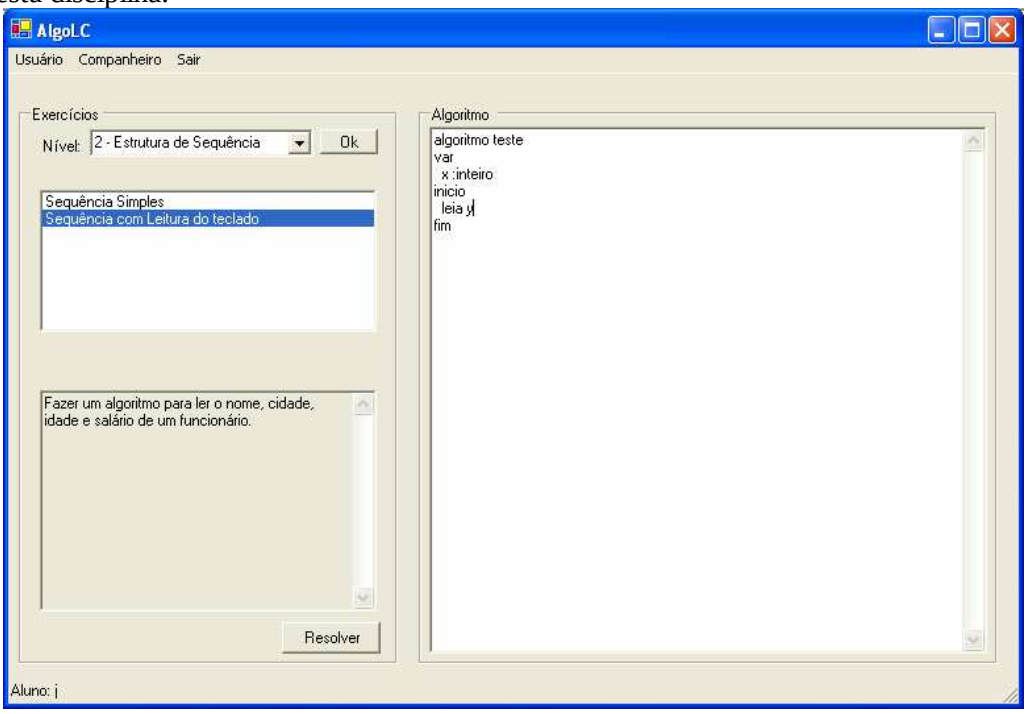

Figura 2 . Interface do AlgoLC

Os exercícios utilizados no teste se restringiram aos de declaração de variáveis e estruturas seqüenciais, totalizando 6 (seis) exercícios. Os testes foram realizados pelo primeiro autor deste artigo durante 3 (três) horas. Primeiramente, os estudantes instalaram o sistema AlgoLC em seu computador pessoal, o que levou em torno de 30 (trinta) minutos. Em seguida, os estudantes tiveram uma breve explicação sobre o sistema e começaram a fazer os exercícios que estavam disponíveis, seguindo a ordem: declaração de variáveis e estruturas seqüenciais. Após a resolução de cada exercício, era solicitado ao estudante enviar a tela de relatório por estudante ao professor. Ao final da resolução de todos os exercícios, o estudante gerava o relatório das restrições, onde se visualizava as estatísticas referentes à violação de cada restrição utilizada em todos os exercícios. A tabela 1 apresenta as restrições que foram violadas. Das 35 (trinta e cinco) restrições modeladas, foram utilizadas 24 (vinte e quatro) restrições. As restrições de 5 a 15 não foram utilizadas durante o teste. As restrições 1, 2, 26, 27, 28 e 29 não foram violadas por nenhum estudante, por isso não constam na tabela. Apenas um estudante violou duas vezes a mesma restrição (restrição 17), que se refere à atribuição de variáveis do tipo inteiro recebendo valores reais.

Dos quinze estudantes que utilizaram o AlgoLC, somente dois conseguiram resolver todos os exercícios sem violar nenhuma restrição. A restrição 23 foi violada 10 (dez) vezes. Essa restrição diz respeito à atribuição de informações à variáveis do tipo *string*, onde a informação a ser atribuída à variável deve estar entre aspas (“ ”). Com a ajuda das mensagens de *feedback* apresentadas pelo CA, os 15 estudantes conseguiram resolver os dois grupos de exercícios propostos.

7. Conclusão e Trabalhos Futuros

Este artigo apresentou o AlgoLC, um SCA que auxilia o ensino e aprendizagem de algoritmos em cursos de graduação em Ciência da Computação. A MBR foi utilizada para modelar o estudante no AlgoLC. O sistema implementa a idéia de que os estudantes podem desenvolver programas de computador através da aprendizagem dos seus erros, mediados por um CA. Ao invés de gerar um modelo de estudante acurado, a MBR permite a identificação dos erros do estudante, que são representados pelo modelo do estudante através das restrições que foram violadas.

No protótipo do AlgoLC foram modeladas e implementadas 35 restrições divididos em 4 grupos de problemas. Do ponto de vista do professor, o AlgoLC facilita o ensino através dos relatórios gerados, onde são identificados os erros dos estudantes. A aprendizagem é auxiliada pela intervenção do CA colaborativ,o que envia mensagens de *feedback* ao estudante. Trabalhos futuros incluem a modelagem de novos problemas e respectivas soluções, assim como a inclusão de estratégias pedagógicas para serem usadas pelo CA.

Tabela 1. Demonstração das violações das restrições pelos estudantes durante o teste.

Estudante	Restrições Utilizadas e Violadas																	
	3	4	16	17	18	19	20	21	22	23	24	25	30	31	32	33	34	35
1																		
2										X		X						X
3		X		X		X			X	X			X			X		
4																		
5	X								X								X	
6										X	X			X				
7				XX		X	X			X					X			
8	X	X	X										X					
9			X		X					X								
10	X									X						X		
11										X								
12		X						X		X				X	X			X
13					X					X								
14		X																
15										X							X	

Referências

Chan, T.W. & Baskin, A.B. Learning Companion Systems. In C. Frasson, & G. Gauthier (eds.), Intelligent tutoring systems: At the crossroads of artificial intelligence and education, Cap. 1, New Jersey, Ablex Publishing Corporation, 1990.

Faraco, R.A., Rosatelli, M.C. & Gauthier, F.A.O. Adaptivity in a learning companion system. In Kinshuk, C.K. Looi, E. Sutinen, D. Sampsom, I. Aedo, L. Uden, & E. Kahkonen (eds.), Proc. of 4th IEEE Int. Conf. on Advanced Learning Technologies, pp. 151-155, Los Alamitos, CA, IEEE CS, 2004a.

Faraco, R.A., Rosatelli, M.C. & Gauthier, F.A.O. An approach of student modeling in a learning companion system. In C. Lematre, C. A. Reyes, & J. A. Gonzalez (eds.), Advances in Artificial Intelligence, pp. 891-900, Berlin: Springer-Verlag, 2004b.

Goodman, B., Soller, A. Linton, F. & Gaimari, R. 1998. Encouraging student reaction and articulation using learning companions. Int. Journal of AIEd 9: 237-255.

Hietala, P. & T. Niemirepo. 1998. The competence of learning companion agents. Int. Journal of AIEd 9: 178-192.

Holt, P., Dubs, S., Jones, M. & Greer, J. The state of student modelling. In J. Greer, & G. McCalla (eds.), Student Modelling:, pp. 3-35, Berlin, Springer-Verlag, 1994.

Kemp, R., Todd, E. & Lu, J.Y. A novel approach to teaching an understanding of programming. In U. Hoppe, F. Verdejo & J. Kay (eds.), Proc. 11th Int. Conf. on AIEd, pp. 449-451, Amsterdam, IOS Press, 2003.

Kim, Y. Pedagogical agents as learning companions: Building social relations with learners. In C.K. Looi, G. McCalla, B. Bredeweg, J. Breuker & H. Pain (eds.), Proc. 12th Int. Conf. on AIEd, pp. 449-451, Amsterdam, IOS Press, 2005.

Martin, B. & Mitrovic, A. automatic problem generation in constraint-based tutors. In S. Cerri, G. Gouarderes, & F. Paraguacu (eds.), Proc. 6th Int. Conf. on ITS, pp. 388-398, Berlin, Springer-Verlag, 2002.

Martin, B. & Mitrovic, A. Domain modelling: Art or science? Artificial Intelligence in Education. In U. Hoppe, F. Verdejo & J. Kay (eds.), Proc. 11th Int. Conf. on AIEd, pp. 183-190, Amsterdam, IOS Press, 2003.

Mitrovic, A., & Ohlsson, S. 1999. Evaluation of a constraint-based tutor for a database language. Int. Journal of. AIEd 10, 238-256.

Ohlsson, S. Constraint-based student modeling. In J. Greer, & G. McCalla (eds.), Student Modeling, pp. 167-189, Berlin, Springer-Verlag, 1994.

Petry, P.G. & Rosatelli, M.C. AlgoLC: A learning companion system for teaching and learning algorithms. In M. Ikeda, K. Ashley, T.W. Chan (eds.), Proc. 8th Int. Conf. on ITS, pp. 775-777, Berlin, Springer-Verlag, 2006.

Rasseneur, D., Delozanne, E., Jacoboni, P.E & Grugeon, B. Learning with virtual agents: Competition and cooperation in AMICO. In S.A. Cerri, G. Gouarderes, & F. Paraguau (eds.), Proc. 6th Int. Conf. on ITS, pp. 61-70, Berlin, Springer-Verlag, 2002.

Regnemalm, E.L. Collaborative dialogue with a learning companion as a cause of information on student reasoning. In Proc. 3rd Int. Conf on ITS, pp. 650-658, Berlin, Springer-Verlag, 1996.

Uresti, J.A.R. Should I teach my computer peer? Some issues in teaching a learning companion. In C. Frasson, G. Gauthier, & K. VanLenh (eds.), Proc. 5th Int. Conf. on ITS, pp. 103-112, Berlin, Springer-Verlag, 2000.

Vizcano, A., Contreras, J., Favela, J. & Prieto, M. An adaptive, collaborative environment to develop good habits in programming. In C. Frasson, G. Gauthier, & K. VanLenh (eds.), Proc. 5th Int. Conf. on ITS, pp .262-271, Berlin, Springer-Verlag, 2000.

Vygotsky, L.S. Mind in Society: The Development of High Psychological Processes, Cambridge, MA, Harvard University Press, 1978.