Composição Dinâmica de Listas de Problemas de Programação Utilizando Algoritmos Genéticos Interativos

Elloá B. Guedes[1], **Mariana R. Nascimento**[1,2], **Francisco G. de Oliveira Neto**[1], **Andréa P. Mendonça**[1], **Gilson P. dos Santos**[1], **Joseana M. Fechine**[1,2]

[1]Laboratório de Inteligência Artificial – Departamento de Sistemas e Computação
Universidade Federal de Campina Grande(UFCG)
Av. Aprígio Veloso, 882, Bodocongó – 58109-970 – Campina Grande – PB – Brasil

[2]Grupo PET Computação (UFCG)

{elloa,mariana,neto,andrea,gilson,joseana}@dsc.ufcg.edu.br

Abstract. *Students of programming courses are constantly dealing with resolution of lists of problems, aiming to consolidate the learning of programming principles and the development of strategies for general problem resolution. These lists are, most of the times, elaborated without any concern regarding the learning difficulties faced by these students, failing in capturing their individual needs of learning. This paper presents an approach that uses interactive genetic algorithms to dynamically compose lists of problems. With this approach, new problems are suggested to the student considering the challenges in solving the previously suggested problem, adapting the problem to be indicated to these student's cognitive level.*

Resumo. *Estudantes de programação lidam constantemente com resolução de listas de problemas a fim de consolidar a aprendizagem dos conteúdos e desenvolver estratégias de resolução de problemas. Na maioria das vezes, a elaboração e proposição destas listas não levam em consideração as dificuldades individuais de aprendizagem dos estudantes. Neste artigo, é apresentada uma proposta para construção dinâmica de listas de problemas, implementada por meio de algoritmos genéticos interativos. Na abordagem proposta, a indicação de um novo problema considera as dificuldades do aluno na resolução de problemas anteriores, adaptando o problema a ser indicado ao nível cognitivo deste estudante.*

1. Introdução

A aprendizagem de programação é majoritariamente pautada na resolução de problemas. A partir desta prática, os estudantes consolidam os conceitos da disciplina, desenvolvem o raciocínio lógico e as estratégias para resolução de problemas.

Para motivar a prática de resolução de problemas, os professores costumam adotar como estratégia pedagógica a aplicação de listas de problemas. Estas listas, na maioria das vezes, agrupam problemas por conteúdo e os ordenam segundo algum critério de complexidade. Entretanto, as listas são propostas igualmente para todos os alunos de uma turma, assumindo que o nível cognitivo dos estudantes é homogêneo. Tal prática incorre em alguns problemas: alunos que não assimilaram o embasamento teórico de forma satisfatória terão dificuldades na resolução dos problemas iniciais, mais fáceis, e□

conseqüentemente estarão impedidos de prosseguir a resolução da lista de forma linear. Por outro lado, alunos com mais conhecimento, sentem-se desmotivados em resolver problemas fáceis, e gostariam de contar com problemas mais desafiadores. Além disso, há casos em que os estudantes sentem dificuldades na resolução de um determinado problema e necessitam resolver problemas similares para solidificar a aprendizagem e estes podem não estar contemplados na lista proposta. Estas situações ocorrem porque as listas de problemas propostas não levam em consideração os aspectos cognitivos dos estudantes e as dificuldades individuais.

Para minimizar esta problemática, é proposta neste artigo uma solução que utiliza algoritmos genéticos interativos, e leva em consideração as dificuldades individuais do aluno na solução de problemas anteriores, a fim de escolher o próximo problema que irá compor a lista.

Este artigo está estruturado conforme descrição a seguir. Na Seção 2, estão descritos os fundamentos teóricos de Algoritmos Genéticos Interativos. Na Seção 3 é apresentada a caracterização da solução proposta. Os resultados obtidos, assim como a avaliação da solução proposta são descritos na Seção 4. Posteriormente, são apresentados os trabalhos relacionados (Seção 5), seguido das considerações finais e sugestões para trabalhos futuros.

2. Fundamentação Teórica

O conceito de Algoritmos Genéticos (AG) foi concebido por Holland e visa a utilização de um paradigma de algoritmos que tem como metáfora o processo evolutivo dos seres vivos, inspirado nos princípios da evolução de Charles Darwin e nos fundamentos da genética [Holland 1975]. O AG é uma representação simplificada do que ocorre na natureza, na qual indivíduos mais bem adaptados ao ambiente sobrevivem à seleção natural e perpetuam (e/ou adaptam) as suas características pelas gerações futuras [Darwin 1859].

Em um AG, as respostas a um problema presente no espaço de busca compõem uma população, e cada problema representa um indivíduo cujas características são descritas em um cromossomo. Cada característica representada no cromossomo denomina-se gene.

Tomando-se como partida uma população inicial, são aplicados operadores que dão origem às novas gerações. Estes operadores representam a seleção natural e farão com que esta população evolua. Em AG, são definidos três operadores:

- Cruzamento: Gera novos indivíduos para a população. Na definição tradicional de AG, o operador de cruzamento seleciona, com determinada probabilidade, dois indivíduos da população e, em seguida, seleciona um ponto aleatório, em cada cromossomo destes indivíduos, e recombina os segmentos parciais definidos por este ponto, gerando um novo cromossomo, processo denominado "*crossover*";
- Mutação: Atua nos cromossomos resultantes do operador de cruzamento, promovendo mudanças em genes escolhidos aleatoriamente. O operador de mutação ocorre com uma probabilidade associada cujo valor é baixo. Caso contrário, a qualidade da população será degenerada e dificilmente haverá convergência do AG para uma solução adequada;
- Seleção: Seleciona os indivíduos resultantes das etapas de cruzamento e mutação que irão compor a nova população, por meio de uma função matemática denominada "função de *fitness*". Existem várias definições para este operador (método da

roleta, seleção por classificação, elitismo, entre outros). No entanto, todas elas visam garantir a maior probabilidade de sobrevivência e reprodução aos organismos mais bem adaptados [Linden 2006].

A definição tradicional do operador de seleção, por intermédio de métodos matemáticos, pode ser pouco eficiente quando há necessidade de incorporação de fatores mais flexíveis na definição de quais indivíduos são melhor adaptados. Algoritmos Genéticos Interativos (AGI) mantêm a idéia original dos algoritmos genéticos, mas introduzem um *feedback* humano na definição deste operador.

O *feedback* humano em um AGI permite a melhor adequação dos resultados do algoritmo a um dado cenário, o qual é difícil de mapear com uma função de *fitness* tradicional [Takagi 1998].

3. Aplicação de Algoritmos Genéticos Interativos na Composição de Listas de Problemas

A opção por uma solução pautada em algoritmos genéticos advém do fato de que a aprendizagem em um determinado domínio corresponde a uma evolução do estado cognitivo do estudante, aspecto representado pelo caráter evolucionário dos algoritmos genéticos. Entretanto, na abordagem clássica de algoritmos genéticos a evolução é marcada, principalmente, por uma função matemática (*fitness*) que geralmente é monotônica. Esta característica é pouco adequada quando se considera um contexto de aprendizagem humana, no qual a evolução nem sempre acontece da mesma forma, ou seja, existem características do mundo que colaboram ou prejudicam a evolução, as denominadas "intempéries".

No contexto educacional, tais intempéries podem ser representadas pelas dificuldades ou pelos *insights* dos alunos no ato da resolução de problemas. Nestes casos, tornam-se necessárias adaptações das estratégias de aprendizagem, por meio da indicação de problemas mais ajustados ao nível cognitivo do estudante. Respeitar o estado cognitivo do aluno consiste em incorporar um fator humano na escolha dos problemas mais adequados, o que caracteriza a aplicação de algoritmos genéticos interativos.

3.1. Caracterização dos Problemas de Programação

No contexto em questão, um problema foi caracterizado considerando os seguintes elementos:

1. Enunciado: Descrição textual do problema;
2. Características do problema: Definidas conforme segue:
   (a) Complexidade: Refere-se ao quão difícil é a resolução do problema, possuindo três classificações: "fácil", "médio" e "difícil";
   (b) Categoria: Relacionada com a natureza do problema, podendo este ser classificado em problema matemático, jogo ou de sistema de informação, por exemplo;
   (c) Assunto: Refere-se ao conteúdo sobre o qual o problema trata, por exemplo, vetores, registros e arquivos;
   (d) Padrões: Referem-se a trechos de código que podem ser reutilizados em outras soluções e correspondem às boas práticas de programação. São considerados dois tipos de padrões: elementares [Delgado 2005] e algorítmicos [Muller et al. 2004]. Padrões elementares são simples,□

sintéticos e baseados na estrutura do código (sintaxe). Referem-se, por exemplo, a operações de seleção, repetição e manipulação de dados. Padrões algorítmicos são soluções básicas que estão ligadas à semântica do problema, por exemplo, ordenação e busca de valores extremos.

Na solução proposta, cada característica descrita acima equivale a um gene na representação cromossômica de um problema. Vale salientar que tais características também constituem-se como métricas comparativas entre problemas. Por exemplo, para o caso de um aluno que tem o primeiro contato com o assunto de vetores é mais indicado iniciá-lo com problemas fáceis e que envolvam poucos padrões, ao invés de problemas com complexidade média e com um número maior de padrões. Do mesmo modo, para um aluno que tenha dificuldade em resolver um problema de matemática, é desejável indicar um novo problema com a mesma complexidade e número de padrões, porém de uma categoria diferente, como jogos ou sistemas de informação.

3.2. Composição Dinâmica de Listas de Problemas

O processo de composição dinâmica de listas de problemas consiste na indicação de um problema ao aluno tomando como referência o resultado (sucesso ou falha) na resolução de um problema anterior, compondo o *feedback* humano característico do AGI. Este *feedback* é capturado por meio de uma ferramenta de submissão de programas, a qual executa um conjunto de testes pré-estabelecidos para avaliar se o programa do estudante atendeu ou não ao que era solicitado.

O processo da composição dinâmica de listas de problemas possibilita ao aluno resolver, para cada assunto da disciplina de programação, um conjunto de problemas com diferentes graus de complexidade e padrões, ajustando-os ao contexto do aluno. Este processo ocorre da seguinte forma: o primeiro passo é determinar qual assunto deve ser aprendido pelo aluno. Dado que ele não resolveu nenhum problema neste assunto, não é possível determinar métricas de dificuldade no aprendizado e, a partir disto, indicar um problema. Neste caso, é selecionado o problema mais fácil, obtido a partir do *cálculo da distância* entre todos os problemas da base para o respectivo assunto e um *problema nulo*.

O cálculo da distância é uma métrica relativa a dois problemas, na qual pondera-se a complexidade e os padrões, para estabelecer o quão difícil é o próximo problema, dado que o anterior foi resolvido. Um problema nulo, por sua vez, é um problema cuja complexidade é definida como "fácil" e que não adota nenhum padrão, representando a ausência de conhecimentos sobre determinado assunto.

A partir do momento que o aluno consegue resolver um primeiro problema, obtido a partir do procedimento descrito, já é possível determinar quais problemas podem ser indicados, levando-se em consideração o nível cognitivo deste aluno, utilizando-se o algoritmo genético para composição da lista de problemas, cujos passos são indicados na figura 1.

Na representação cromossômica de um problema, não foi considerado o seu enunciado, pois vários problemas com cromossomos iguais podem ter enunciados distintos. Portanto, tornam-se equivalentes do ponto de vista do Algoritmo Genético Interativo em questão. Assim, ao levar em conta o termo "problema", considera-se apenas as suas características (complexidade, categoria, assunto e padrões, conforme descritos na Seção 3.1). □

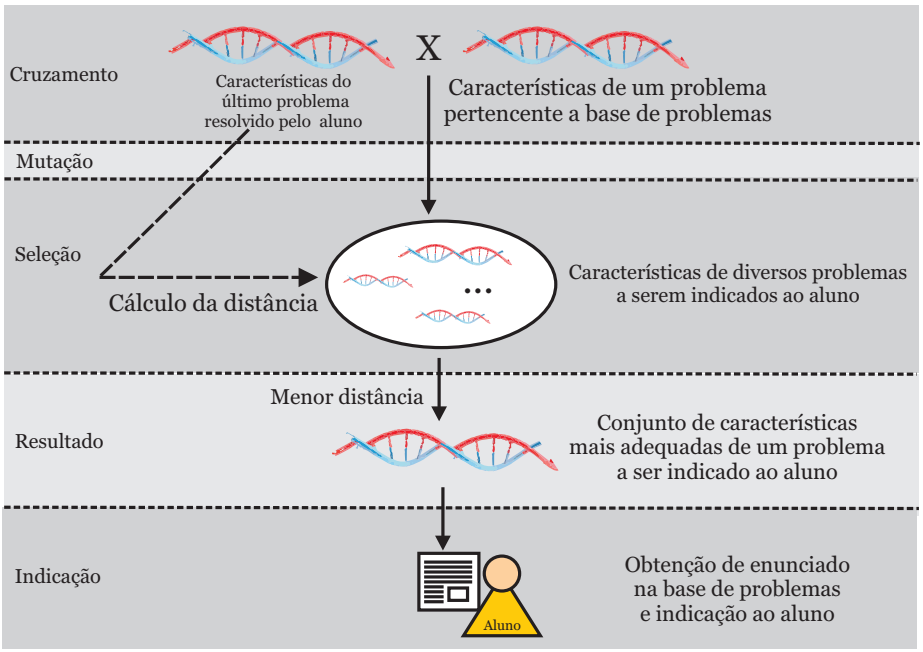

Figura 1. Indicação de um Problema utilizando o Algoritmo Genético Interativo.

Dado o último problema resolvido pelo aluno, o primeiro passo é obter aleatoriamente, da base de dados, um problema de mesmo assunto e efetuar o cruzamento.

Da etapa de cruzamento, é gerado um conjunto de possíveis problemas a serem indicados ao aluno. Este conjunto pode vir a sofrer mutações, de acordo com uma probabilidade associada. A mutação modifica as características do problema alterando a complexidade e os padrões do problema a ser indicado, com base nos problemas que estão sendo cruzados.

Na etapa de seleção, determina-se qual destes problemas deve ser indicado ao aluno. Para tanto, é utilizada a menor distância entre cada problema no conjunto e o último problema resolvido pelo aluno. É de interesse indicar o problema cuja distância foi menor em relação ao último problema resolvido.

Ao término deste processo, o problema deve ser indicado ao aluno (fase de indicação) e agora inclui o enunciado. Dado o sucesso ou falha do aluno na solução deste problema, o ciclo se reinicia até que o aluno atinja a meta definida pelo professor.

Existem duas situações que devem ser consideradas neste processo. A primeira situação ocorre quando um aluno não consegue resolver um problema, o que impede a sua progressão na resolução da lista. Desse modo, para evitar um avanço ou retrocesso prematuros, antes da decisão de indicar um problema mais fácil ou mais difícil, são indicados dois problemas com características iguais as do problema que o aluno não obteve sucesso na resolução.

A segunda situação leva em consideração limitações da base de dados, a qual pode não conter enunciados de problemas com as características exigidas para indicação. Nesta situação, uma nova seleção deve ser realizada, pois outro problema adequado ao nível de conhecimento do aluno pode estar presente na base de dados e deve ser indicado. □

4. Avaliação da Solução Proposta

A avaliação do AGI foi planejada em duas fases: a primeira por meio de testes automatizados e a segunda com alunos em um contexto de sala de aula. Nas próximas seções serão descritos o cenário de avaliação e os resultados obtidos na primeira fase. A avaliação no contexto de sala de aula será realizada em trabalhos futuros.

4.1. Cenário

A avaliação da solução proposta foi realizada por meio da execução de testes automáticos levando em consideração o seguinte cenário:

1. Problemas com as seguintes características: Três níveis de complexidade (fácil, médio e difícil) e um conjunto de 7 padrões, dentre os quais 4 padrões algorítmicos e 3 padrões elementares;
2. Aprendizado de 4 assuntos da disciplina de programação: Vetores, Ponteiros, Recursividade e Arquivos, respectivamente. A condição para que o aluno passe de um assunto para outro é a resolução de um problema difícil e que adote pelo menos 45% dos padrões. Esta condição não implica que o aluno resolverá apenas um problema difícil, dado que existem problemas com essa mesma complexidade, mas que não adotam a mesma porcentagem de padrões;
3. Três perfis de alunos:
   (a) *Aluno Ideal*: Possui 100% de aproveitamento na resolução dos problemas, ou seja, é capaz de solucionar todos os problemas que lhe são propostos;
   (b) *Aluno Bom*: Possui até 80% de sucesso na resolução de problemas;
   (c) *Aluno Regular*: Acerta, em média, até 50% dos problemas que lhe são propostos.

Para simular estes perfis de aluno foram gerados valores binários pseudo-randômicos indicando o sucesso ou falha do aluno na resolução do problema.

Foram realizados, no total, 750 casos de teste, sendo 250 relativos a cada perfil de aluno considerado. Foi utilizada uma base de dados contendo 1000 problemas de programação abrangendo os assuntos e padrões descritos anteriormente.

4.2. Resultados

A análise dos resultados leva em consideração duas métricas: Média e Desvio Padrão do número de problemas resolvidos, por cada perfil de aluno, para a aprendizagem dos quatro assuntos propostos.

Para o aprendizado dos quatro assuntos, considerando o perfil de aluno *ideal*, foi necessária, em média, a resolução de 56 problemas. O aluno *bom*, para a mesma circunstância de aprendizagem, precisou resolver 72 problemas e o aluno *regular*, 87 problemas, conforme descrição na Tabela 1.

A partir da observação dos dados e, considerando as variações da taxa de acerto definidas para cada perfil de aluno (100% para aluno *ideal*; 80% para aluno *bom* e 50% para aluno *regular*), pode-se perceber que o aumento do número médio de problemas para cada perfil de aluno é proporcional à variação da taxa de erro. Para exemplificar, segundo o cenário de avaliação, um aluno *regular* possui taxa de erro 50% superior a do aluno *ideal* que é de 0%. Assim, o número médio de problemas resolvidos por um aluno *regular* é, aproximadamente, 50% maior do que o do aluno *ideal*. Para constatar tal observação basta realizar o seguinte cálculo: □

Tabela 1. Resultados Obtidos

	Perfil do Aluno		
Critérios	**Ideal**	**Bom**	**Regular**
Média de Problemas	56,238	72,704	87,944
Média de Problemas Vetores	14,188	18,752	25,3
Média de Problemas Recursividade	15,792	21,844	24,044
Média de Problemas Apontadores	13,392	16,884	18,616
Média de Problemas Arquivos	12,956	15,224	19,984
Desvio Padrão	10,812	24,965	36,117

Média de problemas do aluno ideal + (Taxa de erro do aluno regular $\times$ Média de problemas do aluno ideal) $\cong$ Média de problemas do aluno regular.

Assim, para o exemplo em questão, tem-se: $56,328 + (50\% * 56,328) = 84,492$, que é um resultado bastante aproximado do valor real da média do aluno *regular* $(87,944)$. Porém, vale salientar que o valor obtido $(84,492)$ pertence ao intervalo relativo ao número de problemas resolvidos por alunos regulares, considerando o desvio padrão para este perfil de aluno $(24,965)$. Este mesmo comportamento foi observado nos outros perfis, o que fornece evidências de que o AGI em questão respeita as características individuais no processo de aprendizagem.

Entretanto, cabe uma ressalva: para que o AGI atue de forma satisfatória, faz-se necessário uma base de dados grande e heterogênea. A diversidade de problemas é necessária porque do contrário, poderá haver muitas indicações repetidas de problemas. Em se tratando da quantidade, se houver uma base muito pequena, o AGI levará muito tempo realizando cruzamentos até encontrar um problema com as características adequadas para indicação. Testes futuros precisam ser realizados a fim de mensurar uma base adequada de problemas que minimize as limitações citadas.

5. Trabalhos Relacionados

Diversas estratégias têm sido utilizadas para prover assistência individualizada aos estudantes, principalmente, no contexto de STI (Sistemas Tutores Inteligentes). Conati e VanLehn [Conati and VanLehn 2001], por exemplo, empregam redes bayesianas para prover material instrucional adequado ao nível cognitivo do estudante. Aleven et al. [Aleven et al. 2006] propôs um modelo de tutor baseado em regras de produção para prover diferentes tipos de ajuda ao estudante. Neste trabalho, a solução foi idealizada com a utilização de algoritmos genéticos. Este recurso já foi vislumbrado em outras aplicações no domínio de STI, a exemplo do trabalho de Lynch et al. [Lynch et al. 2008], o qual apresenta um algoritmo para classificar argumentação de alunos em um domínio de problemas mal-definidos.

Com a utilização de AG, economiza-se tempo na modelagem, que é bastante dispendiosa, se comparada à utilização de redes bayesianas. Considerando as redes bayesianas existe também um esforço na ponderação e ajuste de seus pesos. Se comparada ao uso de regras de produção, a utilização de AG minimiza dificuldades provenientes da criação e manutenção das regras, à medida que a base de problemas vai se modificando.

Até o momento, não foram encontrados trabalhos com a utilização de AG para a composição dinâmica de listas de problemas. Este trabalho, portanto, contribui para a ampliação do estado da arte e disponibiliza à comunidade mais uma estratégia para assistência individualizada ao aluno. □

6. Considerações Finais

Neste artigo, foi apresentada uma solução baseada em Algoritmos Genéticos Interativos para composição dinâmica de problemas no domínio de programação, tendo por objetivo respeitar as necessidades individuais de aprendizagem dos estudantes.

A solução proposta foi implementada em Java e utiliza JDBC (*Java Database Connectivity*) para comunicação com a base de dados, na qual estão armazenados os problemas. A arquitetura modular possibilita que esta funcionalidade seja facilmente incorporada a ambientes virtuais de aprendizagem. Assim, contribui para o reuso da aplicação e consolidação das estratégias para prover assistência individualizada aos alunos.

Na primeira fase de avaliação, realizada por meio de testes automáticos, verificou-se que o AGI proposto apresenta resultados realistas na proposição de problemas, considerando os perfis de alunos definidos no cenário de testes. Em trabalhos futuros, espera-se realizar a segunda fase de avaliação, com alunos em um contexto de sala de aula. Neste caso, será possível avaliar se os resultados obtidos na primeira fase estão de acordo com o cenário real.

Este trabalho é parte integrante de um projeto de pesquisa envolvendo a construção de um ambiente de apoio à resolução de problemas no domínio de programação.

Referências

Aleven, V., McLaren, B., Roll, I. ., and Koedinger, K. (2006). Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence and Education*, 16:101–128.

Conati, C. and VanLehn, K. (2001). Providing adaptive support to the understanding of instructional material. *Proceedings of IUI 2001, International Conference on Intelligent User Interface. Santa Fe, NM.*, pages 41–47.

Darwin, C. (1859). *A Origem das Espécies*. John Murray.

Delgado, K. V. (2005). Diagnóstico baseado em modelos num sistema tutor inteligente para programação com padrões pedagógicos. Master's thesis, Universidade de São Paulo. Instituto de Matemática e Estatística.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michign Press.

Linden, R. (2006). *Algoritmos Genéticos - Uma importate ferramenta da Inteligência Computacional*. BRASPORT Livros e Multimídia Ltda.

Lynch, C., Ashley, K., Pinkwart, N., and Aleven, V. (2008). Argument graph classification via genetic programming and c4.5. *R. Baker & J. Beck (Eds.) Proceedings of the 1st International Conference on Educational Data Mining. Montreal, Canada.*, pages 67–76.

Muller, O., Haberman, B., and Averbuch, H. (2004). (an almost) pedagogical pattern for pattern-based problem-solving instruction. *SIGCSE Bull.*, 36(3):102–106.

Takagi, H. (1998). Interactive evolutionary computation: System optimization based on human subjective evaluation. In *INES'98*.