Uma Aplicação de Algoritmos Genéticos Interativos para Composição de Listas de Problemas de Programação

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,netojin,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 cognitive level of the 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 according to the student's cognitive level. With this approach, new problems are suggested to the student considering the challenges in solving the previously suggested problem.*

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 os diferentes níveis cognitivos dos estudantes e, dessa forma, não capturam as necessidades individuais de aprendizado. Neste artigo, é apresentada uma proposta para construção dinâmica de listas de problemas, implementada por meio de algoritmos genéticos interativos, que leva em consideração o nível cognitivo dos estudantes. Nesta abordagem, a indicação de um novo problema considera as dificuldades do aluno na resolução do problema anterior.*

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 as quais são propostas igualmente para todos os alunos de uma turma. Esta prática não leva em consideração os aspectos cognitivos dos estudantes e suas 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 da seguinte forma: na Seção 2, estão descritos os fundamentos teóricos de Algoritmos Genéticos Interativos. Na Seção 3 é apresentada a solução proposta, seguida das considerações finais e 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 mudança em um gene escolhido 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. Solução Proposta

A opção por adotar uma solução pautada em Algoritmos Genéticos Interativos (uma versão não tradicional de algoritmos genéticos) deu-se em virtude da necessidade de con-

siderar o *feedback* do aluno na resolução do problema atual como característica relevante para a indicação do próximo problema a ser resolvido. Fato que não pode ser assumido na abordagem clássica de algoritmos genéticos, pois a evolução é marcada, principalmente, por uma função matemática (*fitness*) que geralmente é monotônica.

No contexto em questão, um problema é representado por um cromossomo que possui os seguintes genes:

1. Complexidade: Refere-se ao quão difícil é a resolução do problema, possuindo três classificações: "fácil", "médio" e "difícil";
2. Categoria: Relacionada com a natureza do problema, podendo este ser classificado em problema matemático, jogo ou de sistema de informação, por exemplo;
3. Assunto: Refere-se ao conteúdo que o problema trata, por exemplo, vetores, registros e arquivos;
4. 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 representação cromossômica de um problema não foi considerado o seu enunciado, pois vários problemas com cromossomos iguais podem ter enunciados distintos.

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, 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.

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 às 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. Considerações Finais

Neste artigo, foi apresentada uma solução baseada em Algoritmos Genéticos Interativos para composição dinâmica de problemas no domínio de programação. Esta aplicação foi concebida com o objetivo de proporcionar uma estratégia de indicação de problemas que respeite as necessidades individuais de aprendizagem dos estudantes.

A solução proposta foi avaliada por meio de testes automáticos nos quais verificou-se que o AGI proposto apresenta resultados realistas na proposição de problemas. 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 com os testes automáticos estão de acordo com o cenário real.

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. No caso 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.

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.

Agradecimento

Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM).

Referências

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.

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*.