Algoritmo Genético Aplicado à Geração Automática de Casos de Teste

Adilson de Almeida Neto[1,2,3], **Rodrigo Lisbôa Pereira**[2,3], **Roberto Célio Limão de Oliveira**[1,2,3]

[1]Programa de Pós-Graduação em Engenharia Elétrica (PPGEE),
Universidade Federal do Pará (UFPA)

[2]Laboratório de Computação Bioinspirada (LCBio),
Universidade Federal do Pará (UFPA)

[3]Laboratório de Tecnologias Computacionais (LabTeC),
Universidade Federal Rural da Amazônia (UFRA)

almeidneto@gmail.com, rodrigo.lisboa@ufra.edu.br, limao@ufpa.br

Resumo. *Testes são uma parte essencial do desenvolvimento de software e, neste sentido, este trabalho propõe um framework para comparação de algoritmos na tarefa de geração de casos de teste e posterior comparação desses resultados para Algoritmos Genéticos, Algoritmos Genéticos com Interação Social e Hill Climbing.*

1. Introdução

Em 1975 foi apresentada a ideia da "adaptação em sistemas naturais e artificiais" [Holland 1975] que fundamentalmente consiste em aplicar os princípios da evolução natural para problemas de otimização. John Holland estudou formalmente o fenômeno da adaptação natural e desenvolveu métodos para que seu mecanismo pudesse ser importado para sistemas de computadores, construindo assim os Algoritmos Genéticos (AG ou GA — *Genetic Algorithm*) [Sonkar et al. 2012].

Desde então, o AG é utilizado como meta heurísticas bioinspirada e vem sendo aplicado em diversos estudos de otimização, nos mais variados campos, inclusive na Engenharia. Neste sentido, este trabalho aplicará AG ao problema de geração de casos de teste, objetivando aumentar a cobertura de testes com os dados gerados. Como proposta, será implementada duas variações de AGs e será realizada uma análise comparativa entre os algoritmos. O primeiro AG será conforme o padrão originalmente proposto por John Holland [Holland 1975] e segundo AG será hibiridizado com Interação Social por meio da Teoria dos Jogos (TJ), visando expandir os estudos propostos em [Pereira et al. 2020].

Além desta introdução, este artigo apresenta na seção 1 a justificatica do trabalho, na seção 3 é apresentado um breve estado da arte, na seção 4 é destacado o escopo definido para o trabalho e, por fim, na seção 5 são apresentadas as considerações finais.

2. Justificativa

De acordo com a Engenharia de *Software*, o teste de *software* trata-se de uma fase crucial em um processo de desenvolvimento, onde ocorre o processo de identicação e transformação de defeitos latentes em ações para serem sanadas, de acordo com uma determinada necessidade identificada no *software*. Apesar da natureza trabalhosa e

custosa, não é possível ignorar a fase de teste no ciclo de desenvolvimento do *software*. Esta fase usa em torno de 50 a 60% dos custos atrelados à produção de *software* [Ramler and Wolfmaier 2006] e, segundo [Sommerville 2011], desempenha um papel crucial em qualquer projeto.

Pela sua necessidade, a fase de teste de *software* é considerada uma boa oportunidade para diminuir os custos da produção e, consequentemente, do tempo de desenvolvimento de um *software*. Apesar da complexidade proveniente por esta etapa, aplicar teste de *software* no ciclo de desenvolvimento provê o aumento da qualidade do produto, desde que seja realizada, corretamente, as boas práticas da Engenharia de *Software* [Sommerville 2011].

3. Estado da Arte

Diversos estudos aplicam a interdiscipliaridade do AG com a Engenharia de *Software*. Todavia, dois estudos são destacados. No primeiro, há a aplicação do AG à geração de testes [Sharma et al. 2016]. E no segundo, há a comparação do algoritmo genético com o algoritmo de otimização Hill Climbing e com outros algoritmos presentes na literatura científica [Harman and McMinn 2010]. É importante frisar que nesses dois estudos há a implementação de um AG padrão e de outro AG associado a outro método matemático ou à outra técnica de otimização, o que vai de encontro às principais motivações deste trabalho.

Uma motivação é refrente ao desenvolvimento de uma ferramenta genérica, que será disponível não apenas como um objeto de um estudo pontual e direcionado a uma determinada problemática, mas que poderá ser utilizada por outros pesquisadores que pretenderem desenvolver estudos correlatos. Outra motivação é alusiva a construção do AG associado com uma técnica que visa prover a melhoria do processo de otimização da metaheurística, através da hibridização com a TJ, que será inserida para realizar a interação social no AG. Posteriormente será realizada uma análise, através do teste de *software*, da sua aplicabilidade em alguns problemas que ainda serão identificados.

4. Escopo do Trabalho

O escopo deste trabalho se dá em três partes. A primeira é referente a construção uma ferramenta de testes (*framework*) onde será possível avaliar os casos de teste para uma função genérica, tendo como métrica de qualidade a cobertura alcançada, ou seja, a quantidade de linhas de código percorridas utilizando os casos fornecidos.

A Figura 1 exemplifica o comportamento do *framework* que será construído. De acordo com a figura, o *framework* aceitará uma função a ser testada e as entradas que serão utilizadas, e retornará a cobertura alcançada por estes testes, assim como o detalhamentoo de informações de onde o código não foi percorrido.

A principal diferença entre esta ferramenta e uma ferramenta de cobertura de código usual é que a mesma além de computar as linhas percorridas, também computará porque certos trechos de código não foram adentrados, ou seja, computará as diferenças de valores esperados para que condicionais relevantes fossem alcançados.

A segunda parte do escopo é referente a implementação de três algoritmos para serem utilizados na ferramenta, sendo:

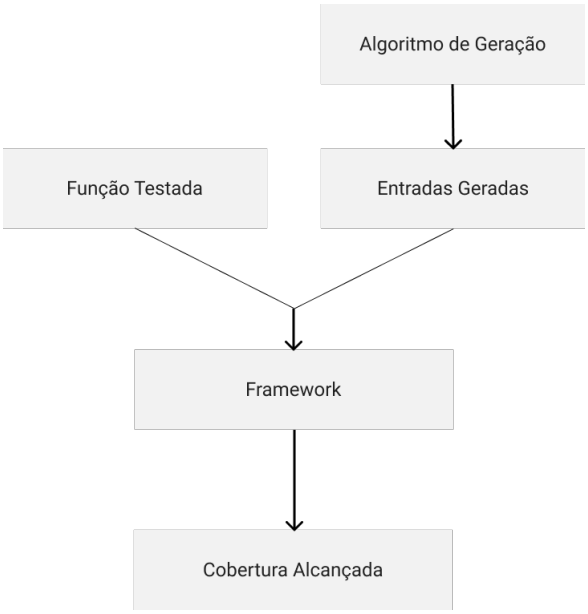

Figura 1. *Framework* de Otimização de Cobertura.

- Algoritmo Genético com Interação Social;
- Algoritmo Genético sem Interação Social;
- Algoritmo Hill Climbing.

A terceira e ultima parte do escopo deste trabalho é referente a avaliação dos três algoritmos, usando como base uma coleção de funções de código livre baseada em [Harman and McMinn 2010], para identificar qual tem a melhor eficácia na geração de casos de teste.

5. Considerações Finais

O trabalho divide-se em três etapas, sendo: 1- Implementação; 2- Definição de problemas; e 3- Análise estatística. Atualmente, o projeto está na etapa 1 e, consequentemente, algumas simulações ainda estão sendo realizadas. Por isso, não foram apresentados resultados. No entanto, os dados da literatura [Harman and McMinn 2010] indicam que esta categoria de abordagem para geração de dados de teste é viável, restando apenas encontrar o equilíbrio entre algoritmos de busca com características mais globais e características locais, e isto será realizado na etapa 2.

Visando a consolidação do estudo, será realizada na etapa 3 uma análise estatística, e através da utilização de algumas métricas indicadas pela literatura, pretende-se corroborar e validar todos os resultados obtidos através das simulações dos algoritmos. Esta etapa irá comparar as métricas de cobertura de teste máximas e médias atingidas à cada problema, em cada algoritmo, para validar a aplicabilidade do algoritmo genético com interação social ao problema e a confiabilidade estatística dos resultados.

Cabe destacar que a aplicação deste trabalho em testes de propriedade é uma área que está sendo estudada pela comunidade científica há alguns anos, onde os algoritmos genéticos já foram usados com sucesso na geração de casos de teste de exceção [Tracey et al. 2000], que podem ser vistos como testes de propriedade onde a propriedade é gerar uma exceção.

Por fim, além da aplicação para a geração dos casos de teste, os algoritmos que serão gerados neste trabalho poderão ser utilizados, também, para a redução de casos de teste baseados em propriedades [Lo et al. 2019]. [MacIver et al. 2019]

Referências

Harman, M. and McMinn, P. (2010). A theoretical and empirical study of search-based testing: Local, global, and hybrid search. *Software Engineering, IEEE Transactions on*, 36:226 – 247.

Holland, J. (1975). An introductory analysis with applications to biology, control, and artificial intelligence. *Adaptation in Natural and Artificial Systems. First Edition, The University of Michigan, USA*.

Lo, F.-Y., Chen, C.-H., and Chen, Y.-p. (2019). Genetic algorithms as shrinkers in property-based testing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 291–292, New York, NY, USA. Association for Computing Machinery.

MacIver, D., Hatfield-Dodds, Z., and Contributors, M. (2019). Hypothesis: A new approach to property-based testing. *Journal of Open Source Software*, 4:1891.

Pereira, R. L., Souza, D. L., Mollinetti, M. A. F., Serra Neto, M. T. R., Yasojima, E. K. K., Teixeira, O. N., and De Oliveira, R. C. L. (2020). Game theory and social interaction for selection and crossover pressure control in genetic algorithms: An empirical analysis to real-valued constrained optimization. *IEEE Access*, 8:144839–144865.

Ramler, R. and Wolfmaier, K. (2006). Economic perspectives in test automation: Balancing automated and manual testing with opportunity cost. pages 85–91.

Sharma, A., Patani, R., and Aggarwal, A. (2016). Software testing using genetic algorithms. *International Journal of Computer Science & Engineering Survey*, 7:21–33.

Sommerville, I. (2011). *Engenharia de software*. Pearson Prentice Hall.

Sonkar, S. K., Malviya, A., Gupta, D., and Chandra, G. (2012). Software testing using genetic algorithm.

Tracey, N., Clark, J., Mander, K., and Mcdermid, J. (2000). Automated test-data generation for exception conditions. *Softw., Pract. Exper.*, 30.