Relato de Experiência do Desenvolvimento de Aplicação da Heurística Algoritmo Genético para Solução do Problema da Próxima Versão

Thiago D. de C. Q. Gama[1], Celso G. C. Junior[2], Gilmar T. Junior[3], Ana Clara A. G. da Silva[3], Ricardo Manuel G. Martins[4]

[1]Núcleo de Tecnologia da Informação
Faculdade de Tecnologia SENAI de Desenvolvimento Gerencial (FATESG) – Goiânia, GO – Brazil

[2]Instituto de Informática (INF)
Universidade Federal de Goiás (UFG) – Goiânia, GO – Brazil

[3]Instituto Acadêmico de Ciências Tecnológicas (IACT)
Universidade Estadual de Goiás (UEG) – Santa Helena de Goiás, GO – Brazil

[4]Escola de Ciências Exatas e da Computação
Pontifícia Universidade Católica de Goiás (PUC Goiás) – Goiânia, GO – Brazil

thiagogama.senai@fieg.com.br, celso@inf.ufg.br,
{gilmar.junior,anaclara.araujo}@ueg.br, ricardomartins@pucgoias.edu.br

Abstract. *This study investigates the application of the Genetic Algorithm heuristic to solve the complex Next Version Problem in software engineering. The problem involves selecting and prioritizing features for the next version of software. The study adapts the Genetic Algorithm to address this issue, demonstrating its effectiveness compared to other configurations through experiments on real data sets. The results indicate that this approach generates efficient and balanced solutions for project objectives, offering valuable insights for requirements management in software development projects.*

Resumo. *Este estudo investiga a aplicação da heurística Algoritmo Genético para resolver o complexo Problema da Próxima Versão na engenharia de software. O problema envolve a seleção e priorização de funcionalidades para a próxima versão de um software. O estudo adapta o Algoritmo Genético para tratar dessa questão, demonstrando sua eficácia em comparação com outras configurações por meio de experimentos em conjuntos de dados reais. Os resultados indicam que essa abordagem gera soluções eficientes e balanceadas para os objetivos do projeto, oferecendo insights valiosos para a gestão de requisitos em projetos de desenvolvimento de software.*

1. Introdução

Este trabalho acadêmico investiga a aplicação da heurística de Algoritmo Genético (AG) como uma abordagem eficaz para resolver o desafiador Problema da Próxima Versão (PPV) na engenharia de software. O Problema da Próxima Versão[1] refere-se à seleção e priorização de requisitos ou funcionalidades a serem incluídos em uma próxima versão de um software, levando em consideração restrições de tempo e recursos humanos.

[1] em inglês, *Next Release Problem* (NRP).

Primeiramente, este estudo apresenta uma revisão abrangente da literatura relacionada à seleção de requisitos, Algoritmo Genético e suas aplicações em engenharia de software. Em seguida, é descrita a modelagem do Problema da Próxima Versão como um problema de otimização multiobjetivo, onde os objetivos incluem a maximização do valor do software, a minimização dos custos e o cumprimento de restrições técnicas e de prazo.

Em seguida, detalha-se a implementação do Algoritmo Genético adaptado para abordar o Problema da Próxima Versão, destacando as representações de solução, operadores genéticos, função de aptidão e critérios de parada específicos. Foram realizados experimentos empíricos usando conjuntos de dados reais e comparativos com técnicas de seleção de requisitos, como outras configurações de parâmetros de entrada da heurística AG e algoritmos de busca exaustiva ou de força bruta (FB).

Os resultados demonstram que o Algoritmo Genético proposto apresenta desempenho considerável em termos de eficácia na seleção de requisitos para a próxima versão do software, gerando soluções eficientes e balanceadas em relação aos objetivos concorrentes. Além disso, discute-se as vantagens, desafios e limitações da abordagem, bem como possíveis direções futuras de pesquisa.

Este estudo contribui para a aplicação prática do Algoritmo Genético na engenharia de software, fornecendo *insights* valiosos para profissionais e pesquisadores que lidam com a gestão de requisitos em projetos de desenvolvimento de software e demonstra a promissora aplicação desta heurística na resolução do PPV.

A subseção 1.1, Motivação, apresenta os fundamentos que impulsionaram a condução da pesquisa, delineando as razões e objetivos que motivam a investigação em questão. Na seção 2, Trabalhos Relacionados, são examinados estudos prévios e trabalhos relevantes que contextualizam e contribuem para o entendimento do tema abordado. A seção 3, Metodologia, descreve a abordagem metodológica adotada para realizar a pesquisa, delineando os passos e procedimentos utilizados. No âmbito da seção 4, focalizando na Modelagem do Problema da Próxima Versão com Algoritmo Genético, a discussão envolve a aplicação de algoritmos genéticos na resolução do problema específico, com ênfase na subseção 4.1, Operadores Genéticos, que explora detalhes operacionais cruciais. A seção 5, Experimentação e Resultados, apresenta os experimentos conduzidos, bem como os resultados obtidos, proporcionando uma visão empírica da aplicação da metodologia. Na seção 6, Discussão, são analisadas as implicações e interpretações dos resultados, proporcionando *insights* sobre os achados. A seção 7, Conclusão e Trabalhos Futuros, resume as descobertas, destaca contribuições significativas e delineia direções futuras para a pesquisa. Finalmente, a seção de Referências lista as fontes bibliográficas utilizadas ao longo do trabalho.

1.1. Motivação

A motivação para este estudo é impulsionada por diversos fatores importantes, entre eles tem-se que o Problema da Próxima Versão é um desafio comum na indústria de desenvolvimento de software. As organizações estão constantemente buscando maneiras de selecionar e priorizar os requisitos de software de forma eficaz para atender às demandas do mercado e aos recursos disponíveis.

A seleção de requisitos para uma próxima versão de software envolve a consideração de múltiplos objetivos e restrições, como custo, prazo, recursos e requisitos técnicos. Isso torna o problema altamente complexo e digno de investigação aprofundada. A aplicação de heurísticas, como o Algoritmo Genético, oferece a oportunidade de encontrar soluções eficientes e balanceadas para o problema, otimizando a alocação de recursos e aumentando o valor das versões do software resultante.

A pesquisa sobre a aplicação do Algoritmo Genético na solução desse problema específico pode preencher lacunas no conhecimento existente em engenharia de software, oferecendo uma nova perspectiva para a seleção de funcionalidades. A aplicação de algoritmos de inteligência artificial, como esse, na resolução de problemas complexos de engenharia de software demonstra uma abordagem inovadora que pode impulsionar avanços consideráveis na área. A eficácia na seleção de requisitos pode levar a economias de custos, redução de desperdícios e melhoria na qualidade do software, fatores críticos para o sucesso de projetos de desenvolvimento de software.

Por fim, o estudo desse problema oferece oportunidades para pesquisadores acadêmicos explorarem conceitos teóricos, adaptando e aplicando o Algoritmo Genético em um contexto prático e relevante. Portanto, a motivação para este trabalho acadêmico está enraizada na necessidade de abordar um problema real e complexo na engenharia de software, bem como na busca por soluções inovadoras e eficazes que possam beneficiar a indústria e a pesquisa acadêmica. Logo, o objetivo desta pesquisa é avaliar a eficácia do Algoritmo Genético na seleção de funcionalidades para a próxima versão do software.

2. Trabalhos Relacionados

Esta síntese de trabalhos relacionados destaca a relevância de Algoritmos Genéticos na solução do Problema da Próxima Versão em engenharia de software, demonstrando sua eficácia na otimização da seleção de funcionalidades em comparação com abordagens tradicionais. Essa base teórica serve como um alicerce sólido para a pesquisa e aplicação prática da heurística Algoritmo Genético nesse contexto. A seguir, é apresentada uma fundamentação teórica com algumas referências relevantes.

A natureza adaptativa e exploratória dos Algoritmos Genéticos os tornam adequados para lidar com as complexas interações entre diferentes requisitos e restrições. Além disso, esses algoritmos podem ser configurados para incorporar preferências do usuário, prioridades de desenvolvimento e outros fatores relevantes para a tomada de decisões, conforme é exposto em [Elvassore 2016].

O Problema da Próxima Versão é um desafio complexo enfrentado no desenvolvimento de software, onde os desenvolvedores precisam decidir quais funcionalidades, correções e melhorias devem ser incluídas na próxima versão de um produto de software. Essa decisão envolve equilibrar requisitos técnicos, restrições de tempo e recursos, prioridades do cliente e objetivos estratégicos da organização.

Algoritmos Genéticos são uma classe de algoritmos de otimização inspirados no processo de seleção natural. Eles têm sido amplamente utilizados em problemas de otimização complexos. [Goldberg 1989] define Algoritmos Genéticos como

“procedimentos de busca baseados em população que evoluem soluções por meio de recombinação genética, seleção de pais, mutação e seleção natural”.

Algoritmos Genéticos têm sido aplicados com sucesso em diversos domínios da engenharia de software. No contexto da seleção de requisitos, [Niu et al. 2008] descrevem a aplicação de Algoritmos Genéticos para otimizar a seleção de recursos em sistemas de software, destacando sua eficácia na busca de soluções eficientes e balanceadas.

O Problema da Próxima Versão é um desafio crítico em engenharia de software, onde é necessário selecionar e priorizar os requisitos que serão incorporados em uma próxima versão de software. [Sommerville 2011] destaca a importância da seleção de requisitos na gestão de projetos de software e a necessidade de considerar restrições de recursos e objetivos de negócios.

Além de Algoritmos Genéticos, várias abordagens tradicionais são utilizadas na seleção de requisitos. Entre elas, métodos baseados em heurísticas e análise multicritério são amplamente reconhecidos. [Gorschek et al. 2006] fornecem uma visão geral dessas abordagens e suas limitações.

3. Metodologia

A seguir, é relatada ordenadamente a metodologia que foi aplicada neste estudo:

1. Inicialmente, descreveu-se detalhadamente o Problema da Próxima Versão em engenharia de software, incluindo suas características, desafios e objetivos;
2. Especificou-se claramente o objetivo da pesquisa;
3. Identificou-se as fontes de dados necessárias para conduzir o estudo, como: registros de requisitos, dados de projetos anteriores e informações sobre restrições de recursos e prazos;
4. Descreveu-se como os requisitos serão representados como indivíduos em uma modelagem de Algoritmo Genético, levando em consideração as características específicas do problema;
5. Foram detalhados os operadores genéticos que serão utilizados, incluindo cruzamento (*crossover*), mutação e seleção, e explicou-se como eles serão aplicados ao contexto do Problema da Próxima Versão;
6. Definiu-se uma função de aptidão que quantifica o desempenho das soluções em relação aos objetivos do projeto, neste caso, maximizar o valor da *release* a partir das *features* disponíveis;
7. Foram especificados os parâmetros do Algoritmo Genético, como tamanho da população, taxa de cruzamento, taxa de mutação e critérios de parada. As escolhas foram justificadas com base na bibliografia especializada abordada neste estudo;
8. Planejou-se os experimentos que serão realizados para avaliar o desempenho do Algoritmo Genético. Isso inclui a execução de simulações em conjuntos de dados reais, que não puderam ser abordados com profundidade devido questões de confidencialidade, e a comparação com outras execuções aplicando parâmetros diferentes e com a execução de um algoritmo de força bruta;

9. Especificou-se as métricas que serão usadas para avaliar o desempenho do AG, como a qualidade das soluções encontradas, o tempo de execução e a análise da convergência dos resultados;
10. Descreveu-se como os resultados serão analisados, incluindo a interpretação das métricas de avaliação e a discussão das implicações dos resultados.

4. Modelagem do Problema da Próxima Versão com Algoritmo Genético

A modelagem do Problema da Próxima Versão com Algoritmo Genético envolve a representação dos requisitos como indivíduos, a definição de operadores genéticos e a formulação da função de aptidão. Abaixo, é apresentada uma modelagem conceitual desse problema (ver Figura 1). A Figura 1, o diagrama de classes da aplicação abordada neste estudo, pode ser verificada no link: https://shorturl.at/erHPX.

Acerca da representação do indivíduo, cada indivíduo (ou solução) apresenta uma lista de genes inteiros e contém uma variável aleatória. Cada solução candidata, que representa uma seleção de requisitos para a próxima versão do software, é codificada como um indivíduo composto por genes. Cada gene pode representar uma funcionalidade específica e seu estado (incluso ou não incluso na próxima versão). Por exemplo, supondo que se tem as seguintes funcionalidades para um software: *feature* A, *feature* B, *feature* C e *feature* D.

Com relação à representação da população, a população contém uma lista de indivíduos e um método que calcula o total de aptidão da população. Podemos representar uma solução candidata como um indivíduo contendo um vetor binário de genes, cada gene desse vetor indica se o requisito correspondente está ou não incluso na próxima versão. Por exemplo, "1010" indicaria que as *features* A e C estão incluídas, enquanto B e D não estão.

No tocante à representação dos desenvolvedores, têm-se os seguintes atributos listados: nome, disponibilidade, nível do desenvolvedor[2], lista de funcionalidades. Sendo que a representação dos níveis dos desenvolvedores é necessária para o cálculo da provisão da equipe de desenvolvedores, a fim de aferir qual é a quantidade de funcionalidades que esse time consegue desenvolver no decorrer da *sprint*. A equação utilizada para calcular a provisão de cada *sprint* é:

$$fp(x) = \sum_{i=0}^{n} p(d).t(d)$$

A quantidade de desenvolvedores da equipe é n. A capacidade de provisão de cada desenvolvedor, d, é p e t é a disponibilidade de tempo em horas semanais que cada desenvolvedor dispõe na *sprint*. O critério de parada usado é a quantidade de gerações.

A seguir são identificados os atributos empregados na representação das funcionalidades, são eles: id, sistema/módulo, projeto/módulo, número da hierarquia do projeto, hierarquia do projeto, tipo, situação, título, desenvolvedor para a qual a *feature*

[2] Os níveis dos desenvolvedores e seus respectivos fatores de multiplicação empregados neste estudo são: júnior (1), pleno (2) e sênior (3).

foi atribuída, catálogo, HET[3], quantidade de serviços, início, tarefa pai, quantidade de anexos, pontos de função, nível de prioridade (seguem os níveis de prioridade das *features* e suas respectivas pontuações aplicadas neste trabalho: urgente (270), alta (90), média (30) e baixa (10)), atribuída e usada, sendo que esses dois últimos atributos citados indicam estados que variam entre os valores "verdadeiro" e "falso".

Nesta listagem é apresentada a descrição dos métodos do indivíduo: **inicializar:** insere no vetor de genes os valores 0 e 1 de forma aleatória; **calcularRestricao:** verifica se o atributo "valido" é verdadeiro, caso negativo chama o método "reparar" em *loop* até que o indivíduo seja válido; **reparar:** muda um gene 1 para 0 em uma posição aleatória; **funcaoObjetivo:** quantifica a aptidão do indivíduo; **getAptidao:** faz um somatório de todas as funcionalidades utilizadas pelo indivíduo, levando em consideração a função de aptidão; **getTotalPontoFuncao:** retorna o somatório dos pontos de função; **mutar:** alterna o gene de 0 para 1 ou vice-versa, conforme a taxa de mutação passada via parâmetro; **isValido:** se o somatório de ponto de função for menor ou igual ao provisionamento retorna válido. Estas são algumas informações específicas da implementação do software tratado neste trabalho.

- **Parâmetros de entrada para a aplicação proposta com seus respectivos exemplos de valores/formatos de entrada:** um arquivo com extensão .csv contendo a listagem de *features* e desenvolvedores disponíveis (colunas: #, sistema-módulo, projeto-módulo, # projeto hierarquia, projeto hierarquia, tipo, situação, título, atribuído para, catálogo, HET (real), qtd. de serviços, início, tarefa pai, qtd. de anexos); taxa de mutação (ex. 0,07); tamanho da população (ex. 100); quantidade de gerações (ex. 1000); chance de cruzamento (ex. 85%); tipo de cruzamento (opções: ponto de cruzamento ou máscara); tipo de seleção de indivíduo (opções: roleta ou torneio) e tamanho do torneio (ex. 2).
- **Informações geradas e exibidas ao término da execução da aplicação:** população, máximo de gerações, taxa de mutação, tamanho do torneio, tempo, gasto, provisão, aptidão do melhor indivíduo, somatório dos pontos de função, quantidade de *features* por prioridade, chance de cruzamento, quantidade de *sprints*, quantidade de acessos à função objetivo, somatório dos pontos de aptidão do *product backlog*, *features*, benefícios, implementadores, *features* selecionadas para a próxima *sprint* e *features* usadas por *sprint*.

Os passos executados no método **iniciar**, que está presente na classe **NextReleaseProblemService** (ver Figura 1) da aplicação implementada neste estudo: carregue as *features* → carregue os desenvolvedores → calcule a provisão total dos desenvolvedores para a *sprint* → crie a população inicial de indivíduos → calcule a quantidade de *sprints* e a aptidão de cada indivíduo → execute um *loop* para cada *sprint* (execute um *loop* para cada geração (execute um *loop* para cada indivíduo da população (selecione os pais (opções: roleta ou torneio) → *crossover* (opções: um ponto de corte ou máscara) → faça a mutação (opção: *flip*[4]) → gere uma nova população))).

4.1. Operadores Genéticos

[3] Sigla de Horas Efetivamente Trabalhadas.

[4] Operação que alterna um ou mais genes selecionados de 0 para 1 (ou *false* para *true*) e vice-versa.

As técnicas de seleção de indivíduos usados na aplicação proposta são: **roleta:** a ideia é que indivíduos com maior aptidão tenham uma probabilidade maior de serem selecionados, semelhante a como uma roleta, que gira para determinar um vencedor; **torneio:** é uma técnica de otimização inspirada pelo processo de seleção natural, aqui se objetiva favorecer os indivíduos mais aptos para que suas características positivas sejam transmitidas às gerações subsequentes, sendo o tamanho do torneio aplicado nos experimentos apresentados neste estudo é igual a 2.

As técnicas de cruzamento implementadas na solução apresentada (em inglês, *crossover*[5]): neste estudo o cruzamento utiliza as seguintes técnicas: **um ponto de corte:** nesta técnica, que se refere ao tamanho do vetor de genes, os genes do filho são criados com os genes do primeiro pai até o ponto de cruzamento e o restante dos genes é coletado daquele ponto de cruzamento em diante do segundo pai; **máscara:** faz referência a um padrão binário que determina quais genes de um indivíduo serão selecionados durante o processo de cruzamento com outro indivíduo para gerar descendentes.

A mutação é usada para introduzir pequenas alterações aleatórias em um indivíduo. Ela pode representar a adição ou remoção de requisitos da solução. Neste estudo, a técnica de mutação empregada é a ***flip***, ela se baseia em trocar o valor do gen de 1 para 0 ou 0 para 1, atendendo à taxa de mutação.

A função de aptidão avalia a qualidade de uma solução candidata em relação aos objetivos do projeto. Nesse estudo, a função de aptidão considerou os múltiplos critérios: **prioridade:** a soma dos valores (importâncias) das *features* selecionadas. Cada requisito pode ter um peso associado que reflete sua importância para os *stakeholders*; **complexidade:** o custo estimado da implementação dos requisitos selecionados, levando em consideração o esforço de desenvolvimento, recursos necessários e custos associados; **atendimento a prazos e recursos humanos disponíveis:** verificação se a seleção de *features* se adequa aos prazos estabelecidos e à força de trabalho disponível para o projeto.

A função de aptidão pode ser formulada como uma combinação ponderada desses critérios, onde o objetivo é maximizar o valor do software enquanto se mantém dentro das restrições de custo, técnica e prazo. A função de aptidão, $fa(x)$, usada na aplicação proposta neste trabalho está descrita abaixo.

$$fa(x) = PF(f).P(f)$$

A função de aptidão consiste na aptidão do indivíduo é feita pelas somatórios de todas as *features* associadas ao indivíduo, referente aos seguintes valores, sendo que $PF(f)$ representa o ponto de função da *feature* e $P(f)$ significa a pontuação de prioridade da *feature.* Essa é uma modelagem do Problema da Próxima Versão usando Algoritmos Genéticos para esta aplicação específica. A complexidade real pode variar dependendo das nuances do problema e dos requisitos específicos do projeto. É importante ajustar essa modelagem com base em detalhes adicionais e realizar experimentos para determinar os parâmetros e configurações ideais do AG.

[5] É usado para criar novas soluções a partir da combinação de dois indivíduos pais.

Gráficos produzidos ao final da execução: **Gantt:** gerado a partir da listagem das *features* por *sprint*. Dado que nesta modelagem cada *sprint* possui 30 dias, as *sprints* são distribuídas neste intervalo de tempo, respeitando a data de inicialização da primeira *sprint*; ***Burndown*:** este mostra o total de *features* ainda não consumidas nas *sprints*.

5. Experimentação e Resultados

Um estudo de caso real foi conduzido em um projeto de desenvolvimento de software, onde o Algoritmo Genético foi aplicado na seleção de requisitos para a próxima versão. Os resultados práticos exibidos na Tabela 1 confirmaram a eficácia da abordagem, sendo que as siglas exibidas nesta tabela (FU, PA, PF e TAP) significam, respectivamente: somatório das *features* utilizadas, somatório dos pontos de aptidão da *sprint*, somatório dos pontos de função da *sprint* e taxa de aproveitamento da provisão na *sprint*.

As configurações do computador e das tecnologias usadas no experimento são: processador Intel(R) Core(TM) i7-5500U - CPU 2.40GHz; memória RAM de 16 GB; sistema operacional: Windows 10; *back-end*: Spring Boot; linguagens de programação e suas respectivas versões: Java 17 e Python 3.11.4; *frameworks* de *front-end*: Thymeleaf e Bootstrap; gerenciador de *build* e dependências: Maven; ambiente de desenvolvimento integrado (em inglês, IDE): IntelliJ IDEA *Community*.

Tabela 1. Dados obtidos no experimento realizado por método de seleção.

Sprint	Seleção	População	Geração	Tempo	∑FU	∑PA	∑PF	TAP
1	Roleta	5	5	3s	156	33880	800	100,00%
	Torneio	5	5	5s	166	34080	798	99,75%
	Roleta	100	1000	1508s	157	39280	800	100,00%
	Torneio	100	1000	185s	168	42810	799	99,88%
2	Roleta	5	5	3s	140	30420	796	99,50%
	Torneio	5	5	5s	149	31190	797	99,62%
	Roleta	100	1000	1508s	153	31360	796	99,50%
	Torneio	100	1000	185s	142	28830	799	99,88%
3	Roleta	5	5	3s	156	30240	792	99,00%
	Torneio	5	5	5s	135	28180	800	100,00%
	Roleta	100	1000	1508s	147	24520	800	100,00%
	Torneio	100	1000	185s	156	24000	800	100,00%
4	Roleta	5	5	3s	156	23910	799	99,88%
	Torneio	5	5	5s	159	25380	800	100,00%
	Roleta	100	1000	1508s	137	24000	800	100,00%
	Torneio	100	1000	185s	149	24000	800	100,00%

5	Roleta	5	5	3s	138	23290	797	99,62%
	Torneio	5	5	5s	143	23110	797	99,62%
	Roleta	100	1000	1508s	159	23940	800	100,00%
	Torneio	100	1000	185s	139	23480	800	100,00%
6	Roleta	5	5	3s	18	2070	83	10,38%
	Torneio	5	5	5s	12	1870	75	9,38%
	Roleta	100	1000	1508s	11	710	71	8,88%
	Torneio	100	1000	185s	10	690	69	8,62%

Na Figura 2 são mostrados os diagramas de engenharia de software gerados, empregando o *framework* JFreeChart, ao término da execução do Algoritmo Genético:

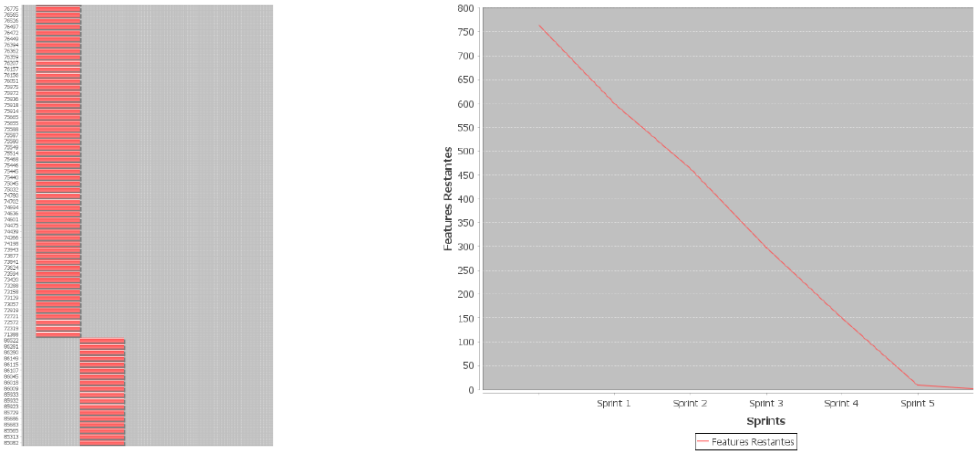

Diagrama de Gantt Diagrama de *Burndown*

Figura 2. Diagramas de engenharia de software gerados pelo sistema referido.

A Tabela 2 mostra a comparação das execuções do AG com os parâmetros: população 100 e gerações 1000 com o algoritmo de FB (obs.: o total de ∑PA é 143810):

Tabela 2. Dados obtidos no experimento realizado por método de seleção.

Sprint	**1 (∑PA)**	**2 (∑PA)**	**3 (∑PA)**	**4 (∑PA)**	**5 (∑PA)**	**6 (∑PA)**
Roleta AG	39280	31360	24520	24000	23940	710
Torneio AG	42810	28830	24000	24000	23480	690
Força Bruta (FB)	48300	24000	24000	24000	22840	670

6. Discussão

À medida que a quantidade de gerações aumenta, o AG melhorou a aptidão dos indivíduos. As métricas de avaliação da aptidão dos indivíduos (tempo de execução, taxa de aproveitamento da provisão e somatório da pontuação de aptidão da melhor

solução encontrada) melhoraram progressivamente, indicando que o algoritmo está escolhendo funcionalidades mais relevantes para a próxima versão e observando as restrições impostas. Isso demonstra a capacidade do Algoritmo Genético de otimizar a seleção de características para melhorar o desempenho do modelo preditivo.

O AG demonstrou uma otimização significativa na seleção de *features* valiosas para a próxima versão do software em comparação com execuções do mesmo com os parâmetros "Gerações" e "População" menores. Isso foi evidenciado pela capacidade do AG de encontrar soluções que otimizam a relação entre os objetivos do projeto, no caso, a maximização do valor das funcionalidades que serão desenvolvidas na próxima *sprint*.

A análise da convergência do AG mostrou que o algoritmo, ao longo das gerações, foi capaz de encontrar soluções ótimas ou próximas do ótimo em um número relativamente pequeno de iterações, economizando tempo de processamento. Dado que, por exemplo, com parâmetros inferiores (pop.: 5; ger.: 5) a melhor solução encontrada na 1ª *sprint* usando o método de seleção roleta alcançou 86,25% do valor do somatório da pontuação de aptidão obtido com parâmetros muito maiores (pop.: 100; ger.: 1000).

7. Conclusão e Trabalhos Futuros

Neste trabalho, constatou-se que a eficácia do AG no PPV depende de uma função de aptidão bem ajustada, aplicando métricas-chave (prioridade e complexidade da *feature*).

Como trabalhos futuros, se almeja realizar as seguintes melhorias neste trabalho: usar mais dados acerca das *features* (por ex.: precedência de *features*) e dos desenvolvedores (por ex.: perfil e produtividade) para aferir a aptidão de cada indivíduo de forma mais precisa e condizente com o que ocorre na realidade; consideração do critério da conformidade com restrições técnicas[6] no cálculo da função objetivo; aplicação de técnicas avançadas de gerenciamento de tempo na solução resultante da execução do Algoritmo Genético; e realizar mais execuções na fase de experimentação a fim de apurar mais dados e, assim, diminuir o viés dos resultados obtidos neste estudo.

A aplicação proposta foi projetada de forma clara para que outros pesquisadores possam replicá-lo, sendo que ela pode ser acessada pela URL https://shorturl.at/lsPUY.

Referências

Elvassore, V. (2016). "Experimenting with generic algorithms to resolve the next release problem". Dissertação de Mestrado. Universitat Politècnica de Catalunya.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley.

Gorschek, T., Wohlin, C., & Östberg, O. (2006). "A staged model for systematic review". In *Empirical Software Engineering*, 11(4), 543-562.

Niu, N., Huang, C., & Jin, H. (2008). "An evolutionary algorithm for feature selection based on mutual information". In *Information Sciences*, 178(14), 2799-2813.

Sommerville, I. (2011). Software Engineering (9th ed.). Addison Wesley.

[6] Avaliação da viabilidade técnica das escolhas feitas, garantindo que os requisitos selecionados sejam compatíveis com a arquitetura e a tecnologia existente no software.