tecnia revista de educação, ciência e tecnologia do ifg

v. 10, Edição Especial 1 | 2025

Submetido 11/06/2024. Aprovado 10/03/2025
Avaliação: revisão duplo-anônimo

Aplicação da meta-heurística algoritmo genético na solução do Problema da Próxima Versão com modelagem, implementação e análise comparativa

THE GENETIC ALGORITHM META-HEURISTIC IS APPLIED TO SOLVE THE NEXT VERSION PROBLEM, AND THE PROCESS IS MODELED, IMPLEMENTED, AND COMPARED

APLICACIÓN DE LA METAHEURÍSTICA DEL ALGORITMO GENÉTICO EN LA SOLUCIÓN DEL PROBLEMA DE LA PRÓXIMA VERSIÓN CON MODELADO, IMPLEMENTACIÓN Y ANÁLISIS COMPARATIVE

Ana Clara A. G. da Silva
Universidade Estadual de Goiás (UEG)
anaclara.araujo@ueg.br

Celso G. C. Junior
Universidade Federal de Goiás (UFG)
celso@inf.ufg.br

Gilmar T. Junior
Universidade Estadual de Goiás (UEG)
gilmar.junior@ueg.br

Ricardo Manuel G. Martins
Pontifícia Universidade Católica de Goiás (PUC Goiás)
ricardomartins@pucgoias.edu.br

Thiago D. de C. Q. Gama
Faculdade de Tecnologia Senai de Desenvolvimento Gerencial (Fatesg)
thiagoddcqg@gmail.com

Resumo

Este estudo apresenta uma investigação sobre a aplicação da meta-heurística algoritmo genético para resolver o complexo Problema da Próxima Versão na engenharia de software. O algoritmo genético foi adaptado 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.

Palavras-chave: algoritmo genético; problema da próxima versão; gerenciamento de requisitos; meta-heurísticas; engenharia de software baseada em busca.

Abstract

This study explores the application of the Genetic Algorithm metaheuristic to address the complex Next Release Problem (NRP) in software engineering. The proposed approach adapts the Genetic Algorithm

to the specific characteristics of this problem and evaluates its performance through experiments conducted on real datasets. The results demonstrate that the method produces efficient and well-balanced solutions aligned with project objectives, providing valuable contributions to requirements management in software development.

Keywords: genetic algorithm; next release problem; requirements management; metaheuristics; search-based software engineering.

Resumen
Este estudio presenta una investigación sobre la aplicación de la metaheurística del algoritmo genético para resolver el complejo problema de la próxima versión en la ingeniería de software. El algoritmo genético fue adaptado para abordar esta cuestión, demostrando su eficacia en comparación con otras configuraciones mediante experimentos en conjuntos de datos reales. Los resultados indican que este enfoque genera soluciones eficientes y equilibradas para los objetivos del proyecto, ofreciendo valiosos conocimientos para la gestión de requisitos en proyectos de desarrollo de software.

Palabras clave: algoritmo genético; problema de la próxima versión; gestión de requisitos; metaheurísticas; ingeniería de software basada en búsqueda.

Introdução

Este trabalho acadêmico apresenta uma investigação sobre a aplicação da meta-heurística algoritmo genético (AG) como uma abordagem eficaz para a resolução do desafiador Problema da Próxima Versão (Next Release Problem – NRP) no contexto da engenharia de software. Esse problema consiste na seleção e priorização de requisitos ou funcionalidades a serem incluídos em uma futura versão de um sistema, considerando restrições de tempo, recursos humanos e demais limitações operacionais.

Primeiramente, foi feita uma revisão abrangente da literatura relacionada à seleção de requisitos, ao algoritmo genético e a suas aplicações em engenharia de software. Em seguida, descreveu-se a modelagem do NRP como um problema de otimização multiobjetivo, cujos objetivos incluem a maximização do valor do software, a minimização dos custos e o cumprimento de restrições técnicas e de prazo.

Na sequência, detalhou-se a implementação do algoritmo genético adaptado para abordar o NRP, destacando as representações de solução, operadores genéticos, função de aptidão e critérios de parada específicos. Na investigação foram realizados experimentos empíricos, usando, para isso, conjuntos de dados reais e comparativos associados a técnicas de seleção de requisitos, como também a outras configurações de parâmetros de entrada da heurística AG e algoritmos de busca exaustiva ou de força bruta (FB).

Os resultados demonstraram 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. Com base nesses resultados, discute-se as vantagens, desafios e limitações da abordagem, bem como possíveis direções futuras de pesquisa. Desse modo, acredita-se que o 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. Além disso, demonstra a promissora aplicação desta heurística na resolução do NRP.

A estrutura deste trabalho está organizada da seguinte forma: Introdução, seção onde são apresentados o contexto e o tema da pesquisa; Motivação, subseção cujo objetivo é apresentar os fundamentos que impulsionaram a realização da pesquisa, destacando as razões e objetivos que justificam sua condução; Bases teóricas, seção em que são apresentados estudos prévios e contribuições relevantes que contextualizam o tema e sustentam teoricamente a abordagem adotada; Decisões metodológicas, seção que descreve a metodologia empregada, detalhando os procedimentos adotados para implementação da solução; Modelagem do com algoritmo genético, seção onde é explorada a aplicação do algoritmo genético ao problema, com ênfase na subseção Operadores genéticos, que apresenta os detalhes operacionais da técnica utilizada; Experimentação e resultados, seção que traz os experimentos realizados e os resultados obtidos, fornecendo evidências empíricas sobre o desempenho da abordagem; Discussão, seção em que se analisa os resultados à luz dos objetivos propostos, interpretando os achados e suas implicações; Proposta de nova modelagem matemática para o Problema da Próxima Versão, seção onde é apresentada uma abordagem alternativa baseada em formulação matemática; Análise da nova modelagem para o Problema da Próxima Versão, seção em que se avalia a eficácia da proposta com base em experimentos adicionais; Considerações finais e trabalhos futuros, seção que sintetiza as principais contribuições da pesquisa e propõe direções para investigações futuras; Referências, seção que reúne as fontes bibliográficas utilizadas ao longo do trabalho.

Motivação

O NRP é um desafio comum na indústria de desenvolvimento de software. Com o avanço da tecnologia, as organizações estão constantemente buscando maneiras de selecionar e priorizar os requisitos de software de forma eficaz, para atender às demandas do mercado com 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 do NRP 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.

Bases teóricas

Esta síntese de bases teóricas destaca a relevância de algoritmos genéticos na solução do NRP em engenharia de software, demonstrando sua eficácia na otimização da seleção de funcionalidades em comparação com abordagens tradicionais. Os estudos citados a seguir servem como um alicerce sólido para a pesquisa e aplicação prática da heurística algoritmo genético nesse contexto.

A natureza adaptativa e exploratória dos algoritmos genéticos os torna 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 exposto em Elvassore (2016).

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

Os 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, oferecendo uma abordagem eficaz em situações que envolvem múltiplos objetivos e restrições. Segundo Harman, Mansouri e Zhang (2012), a engenharia de software baseada em busca (Search-Based Software Engineering – SBSE) aplica algoritmos como os genéticos em diversas fases do ciclo de vida do software, incluindo seleção de requisitos, planejamento de projeto, manutenção e reengenharia. Essa abordagem é particularmente atrativa em cenários caracterizados por grandes espaços de solução e múltiplos objetivos conflitantes, como é o caso do NRP, reforçando a escolha dos algoritmos genéticos como estratégia de otimização adequada para esse contexto.

Algoritmos genéticos têm sido aplicados com sucesso em diversos domínios da engenharia de software. No contexto da seleção de requisitos, Niu, Huang e Jin (2008) descrevem sua utilização para otimizar a escolha de funcionalidades em sistemas de software, destacando a busca por soluções eficientes e balanceadas. Ampliando essa perspectiva, Souza e Rebouças Filho (2020) discutem aplicações mais recentes de algoritmos genéticos na engenharia de software, incluindo sua utilização em ambientes de computação distribuída e aprendizado de máquina, o que evidencia a flexibilidade e a robustez dessa meta-heurística em diferentes contextos computacionais.

O NRP é um desafio crítico em engenharia de software, pois exige a seleção e priorização dos requisitos que serão incorporados em uma próxima versão de software. Nesse sentido, 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 dos algoritmos genéticos, diversas abordagens tradicionais têm sido utilizadas na seleção de requisitos, como métodos baseados em heurísticas e técnicas de análise multicritério. Gorschek, Wohin e Östberg (2006) fornecem uma visão geral dessas estratégias e destacam suas limitações quanto à escalabilidade e à capacidade de tratar múltiplos critérios de forma integrada. Complementando essa visão, Sarrab e Al Shibli (2019) realizaram uma revisão sistemática sobre técnicas de otimização no planejamento de lançamentos de software, evidenciando a eficácia dos algoritmos genéticos na priorização de requisitos em cenários complexos. Essa combinação de

estudos reforça a relevância dos algoritmos genéticos como ferramenta de apoio à tomada de decisão em ambientes com restrições multifatoriais.

Decisões metodológicas

A seguir, são relatadas, de forma ordenada, as decisões metodológicas aplicadas neste estudo:

1. Descrição detalhada do NRP em engenharia de software, incluindo suas características, desafios e objetivos.
2. Especificação clara do objetivo da pesquisa.
3. Identificação das fontes de dados necessárias para conduzir o estudo, a saber: registros de requisitos, dados de projetos anteriores e informações sobre restrições de recursos e prazos.
4. Descrição de como os requisitos seriam representados como indivíduos em uma modelagem de algoritmo genético, levando em consideração as características específicas do problema.
5. Detalhamento dos operadores genéticos que seriam utilizados, incluindo cruzamento (em inglês, *crossover*), mutação e seleção, e explicação de como eles seriam aplicados ao contexto do Problema da Próxima Versão.
6. Definição de uma função de aptidão que quantificasse o desempenho das soluções em relação aos objetivos do projeto, nesse caso, maximização do valor da release a partir das *features* disponíveis.
7. Especificação dos 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. Planejamento dos experimentos que seriam realizados para avaliar o desempenho do algoritmo genético. Isso incluiu a execução de simulações em conjuntos de dados reais – que não puderam ser abordados com profundidade devido às questões de confidencialidade –, e a comparação com outras execuções aplicando parâmetros diferentes e com a execução de um algoritmo de força bruta.
9. Especificação das métricas que seriam 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. Descrição de como os resultados seriam analisados, incluindo a interpretação das métricas de avaliação e a discussão das implicações dos resultados.

Modelagem do Problema da Próxima Versão com algoritmo genético

A modelagem do NRP com algoritmo genético envolve requisitos como a representação dos indivíduos, a definição de operadores genéticos e a formulação da função de aptidão.

Acerca da representação do indivíduo, cada um (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), supondo-se, por exemplo, que se tem as seguintes funcionalidades para um software: *feature* A, *feature* B, *feature* C e *feature* D.

Com relação à representação da população, ela contém uma lista de indivíduos e um método que calcula o total de aptidão da população. Pode-se representar uma solução candidata como um indivíduo contendo um vetor binário de genes, sendo cada um desses genes do 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 e lista de funcionalidades. 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 1 utilizada para calcular a provisão de cada *sprint* é:

$$fp(x) = \sum_{i=0}^{n} p(d).t(d) \quad (1)$$

A quantidade de desenvolvedores da equipe é n. A capacidade de provisão de cada desenvolvedor d é p,enquanto 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, quais sejam: id; sistema/módulo; projeto/módulo; número da hierarquia do projeto; hierarquia do projeto; tipo; situação; título; desenvolvedor para o qual a *feature* foi atribuída; catálogo; HET[1]; 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. Tais pontuações são: urgente (270), alta (90), média (30) e baixa (10), atribuídas e usadas. 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 utilizados:

- **inicializar:** insere no vetor de genes os valores 0 e 1 de forma aleatória.
- **Calcular restrição:** 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.
- **função objetivo:** 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 ao somatório dos pontos de função.

1 Horas Efetivamente Trabalhadas: quantidade real de horas registradas como trabalho dedicado à implementação de uma funcionalidade, podendo divergir da estimativa inicial prevista (Departamento Estadual de Trânsito de Goiás, 2020).

- **mutar:** alterna o gene de 0 para 1, ou vice-versa, conforme a taxa de mutação passada via parâmetro.
- **isValido:** se o somatório de pontos de função for menor ou igual ao provisionamento retorna válido.

Algumas informações específicas da implementação da aplicação tratada neste trabalho são:

- **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), quantidade de serviços, início, tarefa pai, quantidade de anexos); taxa de mutação (por exemplo, 0,07); tamanho da população (por exemplo, 100); quantidade de gerações (por exemplo, 1000); chance de cruzamento (por exemplo, 85%); tipo de cruzamento (opções: ponto de cruzamento ou máscara); tipo de seleção de indivíduo (opções: roleta ou torneio); e tamanho do torneio (por exemplo, 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 a seguir descrevem, de forma estruturada e em linguagem natural, o fluxo principal executado pelo método iniciar, localizado na classe NextReleaseProblemService (Figura 1). Esse método conduziu toda a lógica da aplicação proposta neste estudo, desde o carregamento dos dados de entrada até o processo evolutivo, que ocorreu ao longo de múltiplas *sprints* e gerações, utilizando operadores genéticos para encontrar soluções otimizadas para o Problema da Próxima Versão.

Carrega-se a lista de *features* a serem alocadas.
Carrega-se a lista de desenvolvedores disponíveis.
Calcula-se a quantidade total de horas disponíveis dos desenvolvedores para a *sprint*.
Cria-se a população inicial de soluções (indivíduos).
Define-se a quantidade de *sprints* necessárias com base nas restrições e no escopo.
Avalia-se a aptidão de cada solução da população inicial.
Inicia-se o processo evolutivo:
Para cada *sprint*:
a. Para cada geração:
 i. Inicia-se uma nova população vazia.
 ii. Para cada indivíduo da população atual:
  – Selecionam-se dois pais (usando roleta ou torneio).
  – Realiza-se o cruzamento (com ponto de corte ou máscara).
  – Aplica-se a mutação (utilizando a técnica de *flip*).
  – O novo indivíduo gerado é adicionado à nova população.
 iii. A população atual é atualizada com os novos indivíduos.

Figura 1 – Algoritmo genético para solução do Problema da Próxima Versão
Fonte: Elaborado pelos(as) autores(as).

Operadores genéticos

As técnicas de seleção de indivíduos usadas na aplicação proposta foram:

- **Roleta:** a ideia é que indivíduos com maior aptidão tenham uma probabilidade maior de ser selecionados, semelhantemente a uma roleta, que gira para determinar um vencedor.
- **Torneio:** é uma técnica de otimização inspirada pelo processo de seleção natural, cujo objetivo é favorecer os indivíduos mais aptos, para que suas características positivas sejam transmitidas às gerações subsequentes. O tamanho do torneio aplicado nos experimentos apresentados neste estudo foi igual a 2.

As técnicas de cruzamento (em inglês, *crossover*) aplicadas na solução apresentada neste estudo foram:

- **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, enquanto 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 a técnica 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 foi a *flip*, que se baseia em trocar o valor do gen de 1 para 0 ou de 0 para 1, atendendo à taxa de mutação.

A função de aptidão é a técnica que avalia a qualidade de uma solução candidata em relação aos objetivos do projeto. Nesse estudo, a função de aptidão considerou múltiplos critérios, entre eles:

- **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, com o objetivo de maximizar o valor do software, enquanto se mantém dentro das restrições de custo, técnica e prazo. A função de aptidão, Equação 2, usada na aplicação proposta neste trabalho está descrita abaixo.

$$fa(x) = PF(f).P(f) \quad (2)$$

A função de aptidão consiste na aptidão do indivíduo e é feita pelo somatório de todas as *features* associadas ao indivíduo, referente aos valores seguintes. Nessa

função *PF(f)* representa o ponto de função da *feature P(f)* e significa a pontuação de prioridade da *feature*. Essa é uma modelagem do NRP usando algoritmos genéticos para essa 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.

Em seguida gráficos produzidos e suas respectivas descrições acrescentadas ao final da execução, conforme descrito em Sommerville (2011).

- **Gráfico de Gantt:** é gerado a partir da listagem das *features* distribuídas por *sprint*. Neste modelo, cada *sprint* possui uma duração fixa de 30 dias. As *sprints* são sequencialmente distribuídas dentro deste intervalo de tempo, respeitando a data de início da primeira *sprint*. Este gráfico é essencial para visualizar o progresso do projeto, permitindo identificar o tempo previsto para a conclusão de cada *feature* e assegurar que as dependências e restrições de precedência sejam respeitadas ao longo do ciclo de desenvolvimento.
- **Gráfico de *burndown*:** representa o total de *features* ainda não concluídas ao longo das *sprints*. Essa visualização é essencial para acompanhar o progresso do projeto em relação ao tempo, permitindo verificar se a equipe está no ritmo adequado para concluir todas as *features* dentro do prazo estipulado. O gráfico evidencia a quantidade de trabalho restante em cada *sprint*, facilitando o controle do andamento das atividades e a adoção de estratégias para manter o projeto dentro do cronograma.

Experimentação e resultados

Um estudo de caso real foi conduzido por meio de um projeto de desenvolvimento de software em que o algoritmo genético foi aplicado na seleção de requisitos para a próxima versão. Os resultados práticos confirmaram a eficácia da abordagem e seguem expostos na Tabela 1, a seguir, cujas siglas exibidas (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 utilizadas no experimento incluíram um processador Intel(R) Core(TM) i7-5500U, com CPU de 2.40 GHz, 16 GB de memória RAM e o sistema operacional Windows 10. No *back-end*, foi utilizado o *framework* Spring Boot, com as linguagens de programação Java 17 e Python 3.11.4. No *front-end*, foram empregados os *frameworks* Thymeleaf e Bootstrap. O Maven foi adotado como gerenciador de build e dependências, e a IDE utilizada foi o IntelliJ IDEA Community Edition.

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%

Tabela 1 – Dados obtidos no experimento realizado por método de seleção
Fonte: Elaborado pelos(as) autores(as).

Na Figura 2 são mostrados os diagramas de engenharia de software gerados com emprego do *framework* JFreeChart ao término da execução do algoritmo genético:

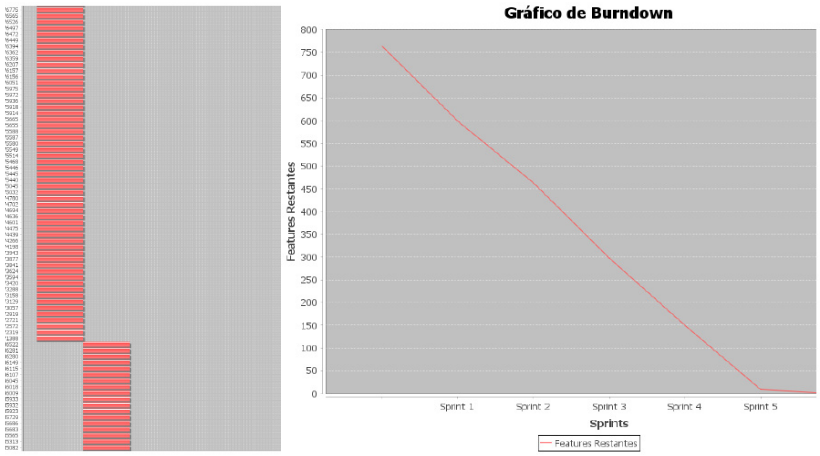

Figura 2 - Diagramas de engenharia de software gerados pelo sistema referido: (a) Diagrama de Gantt, (b) Diagrama de burndown.
Fonte: Elaborado pelos(as) autores(as).

A Tabela 2 apresenta a comparação das execuções do AG com os seguintes parâmetros: população de 100 e de 1000 gerações em relação ao algoritmo de força bruta (FB) (observação: o total de ∑PA é 143810).

Sprint	1 ()	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	48300	24000	24000	24000	22840	670

Tabela 2 – Dados obtidos no experimento realizado por método de seleção
Fonte: Elaborado pelos(as) autores(as).

Discussão

Com base nos resultados obtidos, observou-se que à medida que a quantidade de gerações aumentou, 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) também 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 ainda uma otimização significativa na seleção de *features,* valiosas para a próxima versão do software em comparação com execuções feitas por ele 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 desenvolvidas na próxima *sprint*.

A análise da convergência do AG demonstrou que, ao longo das gerações, ele foi capaz de identificar soluções ótimas ou próximas do ótimo em um número relativamente reduzido de iterações, o que resultou em economia de tempo de processamento. Como evidência, observou-se que mesmo utilizando configurações menos robustas, como população de tamanho 5 e apenas 5 gerações, a melhor solução obtida na primeira *sprint* com o método de seleção por roleta atingiu 86,25% do somatório da pontuação de aptidão alcançada com configurações significativamente superiores, envolvendo população de 100 indivíduos e de 1000 gerações.

Proposta de nova modelagem matemática para o Problema da Próxima Versão

A partir das informações do estudo de Elvassore (2016), diversas melhorias foram propostas para a implementação realizada. O principal motivo para utilizar o estudo de Elvassore como base foi a combinação de AG com o NRP, que é uma área de grande interesse. Além disso, o estudo utilizou o jMetal, um *framework* específico e bem estabelecido nesse campo de investigação. O estudo também fez uso de meta-heurística multiobjetivo de algoritmos genéticos, que são amplamente citadas na literatura atual.

A proposta elaborada envolve uma reformulação do NRP para melhor se adequar aos desafios atuais de pesquisa. Essa reformulação considera as *features* disponíveis e as atribui aos funcionários habilitados. Os principais pontos da proposta incluem considerar o custo como horas humanas e o valor como prioridade, além de implementar um planejamento preciso de *features* a serem desenvolvidas, respeitando restrições de precedência, disponibilidade de recursos humanos, competências e datas finais de desenvolvimento.

O NRP foi modelado considerando diversas variáveis, entre elas: as funcionalidades, entendidas como melhorias demandadas pelos clientes e associadas a custos específicos; as restrições de precedência, que estabelecem dependências entre *features*; os clientes, com seus respectivos valores estratégicos para a organização; os requisitos vinculados a cada *feature*; e um orçamento que não deve ser excedido. A adaptação do modelo proposta neste estudo incluiu: a atribuição de prioridade a cada funcionalidade; a definição de uma lista de funcionários com disponibilidade semanal; e a exigência de que cada *feature* seja executada apenas por profissionais com a habilidade correspondente. Além disso, a data final passou a substituir o orçamento global como restrição principal, com o objetivo de maximizar o número de *features* implementadas (ponderadas por sua prioridade) e, simultaneamente, minimizar a data de conclusão do projeto.

A implementação envolve classes para avaliar objetivos e restrições da solução, contendo *features* planejadas e funcionários. Também utiliza algoritmos de mutação, cruzamento, planejamento de *features* e geração de *features*.

Essa proposta de nova modelagem matemática para o NRP visa otimizar a seleção de *features* a serem implementadas em um software, maximizando o valor de negócio enquanto respeita as restrições de capacidade dos empregados e o esforço total. Nesta pesquisa, um algoritmo genético foi utilizado para explorar o espaço de soluções, modelando *features* e empregados com seus respectivos atributos e relacionamentos. Parâmetros como valor de negócio, esforço necessário, capacidade dos empregados e precedência entre *features* foram considerados, garantindo uma alocação eficiente e priorizada das tarefas.

A modelagem foi formulada com base em elementos matemáticos de otimização combinatória, sendo validada através de uma implementação que integra e executa essas regras. Essa abordagem proporciona uma solução prática e eficaz para a gestão de *releases* de software, permitindo uma melhor alocação de recursos e um planejamento estratégico mais preciso das funcionalidades a serem desenvolvidas em cada versão.

A seguir, apresenta-se a formulação matemática da nova modelagem proposta para o Problema da Próxima Versão.

Parâmetros:

- **Valor de negócio da *feature*:** importância ou benefício que a *feature* proporciona para a organização ou para o cliente.
- **Esforço necessário para implementação:** quantidade de trabalho ou recursos necessários para desenvolver a *feature*.
- **Conjunto de *features*:** conjunto de funcionalidades ou melhorias a serem consideradas para inclusão na próxima versão do produto.
- **Equipe de funcionários:** equipe de desenvolvimento disponível para trabalhar nas *features*.

- **Quantidade máxima de *features*:** limite máximo de funcionalidades que podem ser incluídas na próxima versão.
- **Esforço máximo disponível:** capacidade total de trabalho que a equipe de funcionários pode dedicar ao desenvolvimento das *features*.
- **Capacidade dos funcionários:** quantidade de trabalho que cada funcionário pode realizar no período considerado.
- **Habilidades dos funcionários:** competências e especializações dos membros da equipe que influenciam quais *features* eles podem implementar.
- **Precedência das *features*:** dependências entre as *features*, indicando quais precisam ser concluídas antes de outras.
- **Tipo da *feature*:** categoria ou classificação da *feature*, como *bug fix*, melhoria ou nova funcionalidade.
- **Status da *feature*:** estado atual da *feature*, como planejada, em desenvolvimento ou concluída.
- **Título da *feature*:** nome ou descrição breve da *feature*.
- **Responsável pela *feature*:** membro da equipe responsável por desenvolver a *feature*.
- **Quantidade de serviço da *feature*:** volume de trabalho ou complexidade associado à *feature*.
- **Data de início da *feature*:** data prevista ou efetiva de início do desenvolvimento da *feature*.
- **Data fim da *feature*:** data prevista para a conclusão ou entrega de uma determinada *feature*.
- **Tarefa pai:** tarefa maior ou projeto ao qual a *feature* está relacionada.
- **Quantidade de anexos:** número de documentos ou arquivos anexados à *feature* para suporte ou referência.
- **Prioridade da *feature*:** importância relativa em relação a outras *features*.

VARIÁVEIS DE DECISÃO:

- x_i: binário indicando se a *feature* i será implementada (1 se for implementada, 0 caso contrário).
- y_{ij}: binário indicando se a *feature* será implementada pelo empregado .

FUNÇÃO OBJETIVO:

- $max \sum_{i=1}^{N} V_i x_i$: maximizar o valor de negócio total das *features* selecionadas.

RESTRIÇÕES:

- $y_{ij} \leq x_i \; \forall \, i \in \{1, ..., N\}, \forall j \in \{1, ..., M\}$: capacidade dos empregados.
- $y_{ij} \leq x_i \; \forall \, i \in \{1, ..., N\}, \forall j \in \{1, ..., M\}$: relacionamento entre *features* e empregados.

- $x_i \leq x_{p_i}, \forall\, i \in \{1, \ldots, N\}, \forall\, P_i$: precedência entre *features*.
- $\sum_{i=1}^{N} E_i x_i \leq E_{max}$: limite de esforço total.
- $\sum x_i \leq F_{max}$: limite de número de *features*.

Nessa implementação descrita, várias tecnologias modernas e *frameworks* foram utilizadas para otimizar a execução e a eficácia da modelagem. O jMetal, por exemplo, foi utilizado para a implementação de algoritmos meta-heurísticos multiobjetivo, oferecendo uma base sólida para desenvolver soluções robustas. Diversos algoritmos genéticos, como Multi-Objective Cellular Genetic Algorithm (MOCell), Non-dominated Sorting Genetic Algorithm II (NSGA-II), (Pareto Envelope-based Selection Algorithm II (PESA-II) e (Strength Pareto Evolutionary Algorithm II (SPEA-II), foram escolhidos por sua eficácia em explorar o espaço de soluções e encontrar resultados otimizados.

A modelagem matemática é baseada em elementos de otimização combinatória, permitindo a formulação de modelos que consideram múltiplas restrições e objetivos. Técnicas avançadas de meta-heurísticas foram aplicadas para a exploração do espaço de soluções, considerando múltiplos parâmetros de negócios e técnicos.

Para testar a API, o Postman foi utilizado para fazer requisições e o Heroku para hospedar a API, facilitando o acesso e a execução remota. Além disso, o Oracle APEX foi usado para testes adicionais. A API pode ser testada por meio da URL https://apex.oracle.com/pls/apex/r/thiagoddcqg/nrpag/login, com as credenciais de acesso (usuário: demo, senha: demodemo), disponibilizadas para avaliação da API.

A imagem abaixo (Figura 3) mostra a interface da aplicação NRPAG no Oracle APEX.

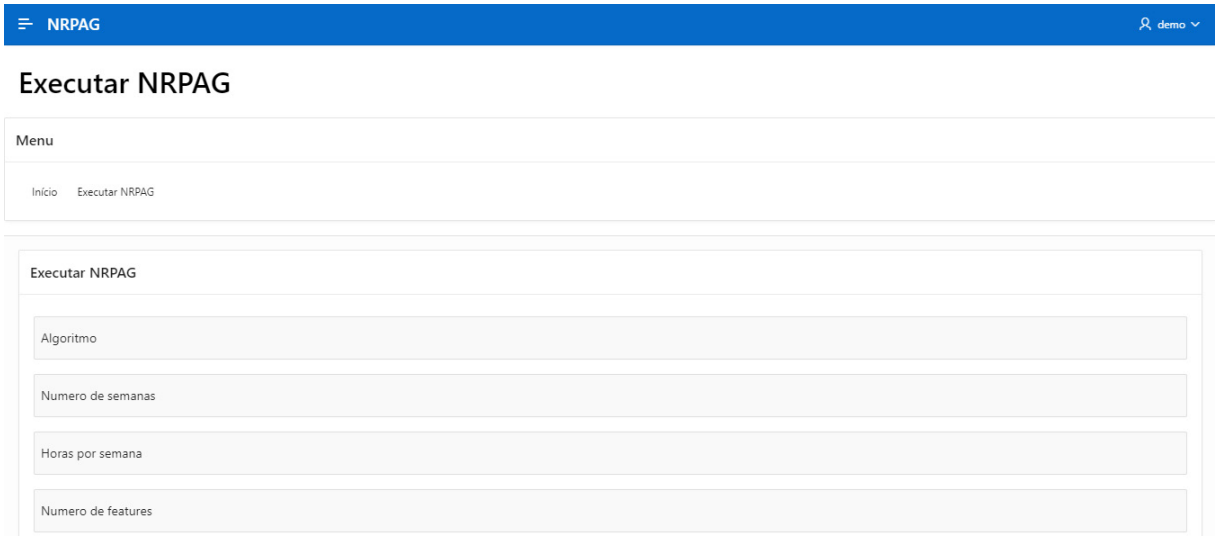

Figura 3 – Interface da aplicação NRPAG na plataforma Oracle APEX
Fonte: Elaborado pelos(as) autores(as).

Análise da nova modelagem para o Problema da Próxima Versão

A implementação aqui apresentada utiliza dados de empresas participantes do projeto Supersede (Projeto […], 2024) para avaliar a qualidade da solução, com pontuações baseadas em prioridade e data final. O projeto Supersede desenvolve ferramentas para a evolução e adaptação contínua de sistemas pervasivos, utilizando feedback de usuários e análise de dados em tempo real. Esses dados fornecem uma base sólida e realista para testar e validar as soluções propostas.

O protocolo do experimento inclui a comparação de diferentes algoritmos com variáveis, como número de *features* e funcionários. Os testes variam o número de *features* e funcionários, considerando restrições de precedência, habilidade requerida e otimização de funcionários. Essas variáveis são fundamentais para entender como cada algoritmo se comporta em cenários diferentes e para identificar as condições sob as quais cada algoritmo é mais eficaz.

Os resultados dos experimentos mostraram que o MOCell foi a solução mais eficaz e rápida para o NRP. Este algoritmo se destacou na capacidade de explorar o espaço de soluções e por encontrar respostas otimizadas rapidamente. Em comparação, NSGA-II e PESA-II apresentaram bons resultados, mostrando-se como alternativas viáveis, dependendo das especificidades do problema.

Por outro lado, o SPEA-II não se mostrou adequado para resolver o NRP proposto. Isso sugere que, apesar de suas capacidades em outras áreas, o SPEA-II pode ter limitações em lidar com as complexidades específicas do NRP quando comparado aos outros algoritmos testados.

Considerações finais

Neste trabalho, constatou-se que a eficácia do algoritmo genético no NRP está diretamente relacionada a uma função de aptidão bem ajustada, que considere métricas-chave como a prioridade e a complexidade das funcionalidades. Com base nos resultados obtidos, são propostas as seguintes melhorias para pesquisas futuras:

- **Utilização de mais dados:** incluir informações adicionais sobre as funcionalidades (como precedência) e sobre os desenvolvedores (como perfil e produtividade), a fim de aferir a aptidão de cada indivíduo de forma mais precisa e alinhada à realidade.
- **Conformidade com restrições técnicas:** incorporar esse critério no cálculo da função objetivo.
- **Gerenciamento de tempo:** aplicar técnicas avançadas de gerenciamento de tempo à solução gerada pelo algoritmo genético.
- **Execuções adicionais:** realizar um maior número de execuções na fase de experimentação para minimizar possíveis vieses nos resultados.
- ***Datasets* consolidados:** utilizar conjuntos de dados consolidados na área e repetir os experimentos diversas vezes, com o intuito de obter resultados estatisticamente significativos.

Essa abordagem proporciona uma solução prática e eficaz para a gestão de releases de software, permitindo uma melhor alocação de recursos e um planejamento estratégico mais preciso das funcionalidades a serem desenvolvidas em cada versão. Outras melhorias incluem a inclusão de mais meta-heurísticas de algoritmos genéticos para explorar diferentes abordagens de otimização.

A eficácia do algoritmo genético no NRP é influenciada por uma função de aptidão bem ajustada, que considera métricas importantes como prioridade e complexidade das *features*. A partir das melhorias propostas, espera-se uma maior precisão na avaliação da aptidão dos indivíduos, levando a soluções mais robustas e aplicáveis na prática.

Para trabalhos futuros, considera-se necessário: ampliar o conjunto de dados sobre as *features* e os desenvolvedores; incluir a conformidade com restrições técnicas; aplicar técnicas avançadas de gerenciamento de tempo; realizar mais execuções experimentais e utilizar *datasets* consolidados; incorporar um maior número de meta-heurísticas de algoritmos genéticos.

Essas melhorias e direções futuras visam aprimorar ainda mais a eficácia e a eficiência da modelagem matemática aplicada ao Problema da Próxima Versão, proporcionando uma ferramenta valiosa para a gestão estratégica de releases de software.

Referências

DEPARTAMENTO ESTADUAL DE TRÂNSITO DE GOIÁS. *Termo de referência* – Anexo I. Goiânia: Detran-GO, 2020. Disponível em: https://goias.gov.br/detran/wp-content/uploads/sites/8/2023/07/20201123-ANEXO-I-TERMO-DE-REFERENCIA.pdf. Acesso em: 30 abr. 2025.

ELVASSORE, Valentin. *Experimenting with generic algorithms to resolve the next release problem*. 2016. Dissertação (Mestrado em Inovação e Pesquisa em Informática - MIRI) — Institut Superieur d'Informatique, de Modelisation et de leurs Applications, Universitat Politècnica de Catalunya, Catalunya, 2016. Disponível em: https://upcommons.upc.edu/entities/publication/e2c6aabe-c3f3-4bef-b8fa-21d3bd45870f. Acesso em: 30 abr. 2025.

GORSCHEK, T.; WOHLIN, C.; ÖSTBERG, O. A staged model for systematic review. *Empirical Software Engineering*, [*s. l.*], v. 11, n. 4, p. 543-562, 2006.

HARMAN, Mark; MANSOURI, Sahar A.; ZHANG, Yuanyuan. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, [*s. l.*], v. 45, n. 1, p. 1-61, 2012. DOI: https://doi.org/10.1145/2379776.2379787. Acesso em: 30 abr. 2025.

NIU, N.; HUANG, C.; JIN, H. An evolutionary algorithm for feature selection based on mutual information. *Information Sciences*, [*s. l.*], v. 178, n. 14, p. 2799-2813, 2008. DOI: https://doi.org/10.1016/j.ins.2015.02.031. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0020025515001310. Acesso em: 30 abr. 2025.

PROJETO SUPERSEDE. *GitHub*, [*s. l.*], c2025. Disponível em: https://github.com/supersede-project. Acesso em: 10 jun. 2024.

SARRAB, M.; AL SHIBLI, I. Optimization techniques for software release planning: A systematic literature review. *Journal of Software: Evolution and Process*, [*s. l.*], v. 31, n. 11, e2153, 2019.

SOMMERVILLE, I. *Software Engineering.* 9. ed. Boston, EUA: Addison Wesley, 2011.

SOUZA, L. F.; REBOUÇAS FILHO, P. P. Applications of genetic algorithm in software engineering, distributed computing and machine learning. *In*: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ITS APPLICATIONS (ICCSA), 2020. *Anais* [...]. Springer, 2020. p. 309-324.