REPODUCIBILITY MINER: AUXILIANDO NA SELEÇÃO DE REPOSITÓRIOS DE SOFTWARE PARA ESTUDOS DE MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

REPODUCIBILITY MINER: SUPPORTING THE SELECTION OF SOFTWARE REPOSITORIES FOR MINING SOFTWARE REPOSITORIES STUDIES

Hiero Henrique Barcelos Costa[1]
Victor Souza Salles[2]
Guilherme Marques de Oliveira[3]
Gleiph Ghiotto Lima Menezes[4]

DOI: 10.34019/2179-3700.2024.v24.46193
ENVIADO EM: 6/10/2024
APROVADO EM: 11/11/2024

PRINCIPIA
CAMINHOS DA INICIAÇÃO CIENTÍFICA
UNIVERSIDADE FEDERAL DE JUIZ DE FORA
EDITORIA DE PÓS-GRADUAÇÃO E PESQUISA (PROPP)
UFJF.BR/PROPP

RESUMO

A hospedagem de repositórios *Git*em plataformas como o *GitHub* transformou-as em tesouros de informações que podem ser utilizadas por pesquisadores em experimentos de Mineração de Repositórios de Software para aprimorar o desenvolvimento de software. Assim, nesse cenário, muitas ferramentas foram criadas para dar suporte a seleção de repositórios. No entanto, elas possuem limitações, pois as abordagens tradicionais utilizadas na seleção de repositórios não apresentam um processo sistemático para escolha dos repositórios e o armazenamento do racional para a justificativa das escolhas, inviabilizando a reprodutibilidade dos experimentos. Portanto, o *RepoDucibility Miner* foi criado com o intuito de resolver os problemas e as limitações de ferramentas apresentadas na literatura. Esse trabalho apresenta o *RepoDucibility*

[1] Acadêmico da Engenharia Computacional da Universidade Federal de Juiz de Fora. Membro do grupo GET - Engenharia Computacional, bolsista do projeto “Seleção de Repositórios” de 2022 a 2023 e membro do grupo de pesquisa Homúnculos. E-mail: hiero.costa@estudante.ufjf.br

[2] Acadêmico do Curso de Ciências Exatas da Universidade Federal de Juiz de Fora. Bolsista de 2023 a 2024 do projeto “Seleção de Repositórios”. E-mail: victor.salles@estudante.ufjf.br

[3] Acadêmico da Ciência da Computação da Universidade Federal de Juiz de Fora. Bolsista do projeto “Seleção de Repositórios” de 2020 a 2021. E-mail: guilhermemarques01@ice.ufjf.br

[4] Professor orientador dos projetos de Iniciação Científica PIBITI. Professor do Departamento de Ciência da Computação da UFJF. E-mail: gleiph@ice.ufjf.br.

Miner e avalia o tempo necessário para realizar diferentes buscas com alterações na configuração da pesquisa. Os resultados demostram que as pesquisar realizadas podem ter um tempo médio de até 20:29:10 e que o aumento dos tokens para 3 pode reduzir o tempo de busca em até 85,51%.

Palavras-chave: Reprodutibilidade de Experimentos. Gerência de Configuração. Mineração de Repositórios de Software. GitHub.

ABSTRACT

The hosting of software programs in Git repositories on platforms such as GitHub turned these platforms into treasure troves of information that can be used by researchers in Mining Software Repositories experiments to enhance software development. Thus, in this scenario, many tools have been created to support repository selection. However, they have limitations since traditional approaches used in repository selection do not present a systematic process for choosing repositories and storing the rationale for justifying the choices, making the reproducibility of experiments unfeasible. Therefore, RepoDucibility Miner was created to solve the problems and limitations of tools presented in the literature. This work presents RepoDucibility Miner and evaluates the time required to perform different searches with changes in the search configuration. The results show that the searches performed can have an average time of up to 20:29:10 and that increasing the number of tokens to 3 can reduce the search time by up to 85.51%.

Keywords: Experiment Reproducibility. Configuration Management. Software Repository Mining. GitHub.

1 INTRODUÇÃO

Os repositórios de software consistem em locais, onde é possível armazenar e recuperar artefatos (*e.g.*, documentação e código-fonte) produzidos durante o ciclo de vida de um software. Esses repositórios podem ser armazenados em plataformas, como *GitHub*[3] e *Bitbucket*[4], que oferecem serviços de hospedagem para programadores e podem ser utilizadas em estudos de Mineração de Repositórios de Software (GÜEMES-PEÑA et al., 2018).

A Mineração de Repositórios de Software (MRS) é um campo que vem ganhando cada vez mais atenção, dada a extensão das informações que podem ser coletadas de repositórios para melhorar as atividades de Engenharia de Software

[3] https://github.com

[4] https://bitbucket.org/

(HASSAN, 2008). Este campo consiste na busca empírica e sistemática de um conjunto de repositórios, para poder analisar as mudanças que ocorrem neles e, assim, descobrir informações e tendências pertinentes no projeto analisado (KAGDI; COLLARD; MALETIC, 2007).

Para facilitar o processo de coleta dos repositórios e extração de seus dados, ferramentas como ModelMine (REZA; BADREDDIN; RAHAD, 2020)*, GHTorrent, G-Repo (ROMANO et al., 2021)* e a própria busca avançada do *GitHub*[5] foram criadas. Entretanto, mesmo com as ferramentas disponíveis hoje, há problemas no processo de seleção e documentação do conjunto de dados utilizado, acarretando problemas de generalização dos resultados presentes na literatura, como visto no estudo conduzido por Vidoni (2022). Segundo Vidoni, apenas 17% apresentam um processo de seleção sistemático.

Este artigo apresenta o *RepoDucibility Miner*, uma ferramenta que visa superar as limitações apresentadas na literatura por meio da fragmentação das buscas, enriquecendo-as com metadados e criando critérios que podem ser associados a cada repositório para armazenar o racional de seleção. Com essas etapas, vislumbra-se auxiliar a documentação e a reprodutibilidade dos experimentos conduzidos, rastreando as decisões tomadas pelo usuário para poderem ser utilizadas em futuras pesquisas. A ferramenta foi avaliada considerando o seu comportamento em duas situações, (1) qual a influência do número de repositórios retornados no tempo de coleta e (2) qual a influência da quantidade de *tokens* utilizados no tempo de pesquisa. Foi observado como o aumento dos repositórios pode resultar em pesquisa de até mais de 20 horas, cenário observado para pesquisas para repositórios que possuem mais de 1.000 estrelas que retornam 49.616 repositórios. Também foi visto como o aumento na quantidade de *tokens* para 3 pode reduzir até 85,51% o tempo de pesquisa.

Este artigo é composto por cinco seções, incluindo a Introdução. A Seção 2 expõe a visão geral do estado da arte em Mineração de Repositórios de Software (MSR), abordando tanto a ferramenta proposta, com suas funcionalidades, quanto as ferramentas que atuam na seleção de repositórios. A Seção 3 apresenta os resultados da análise do comportamento da ferramenta para os dois casos apresentados. A Seção 4 discute as respostas para as questões levantadas na Seção 3. Por fim, a Seção 5 apresenta as principais contribuições e as perspectivas de trabalhos futuros.

[5] https://github.com/search/advanced

2 METODOLOGIA

Como parte da pesquisa MSR, é necessário selecionar os repositórios para conduzir um estudo. Assim, as plataformas de hospedagem são utilizadas para a seleção do repositório, sendo, segundo Luzgin e Kholod (2020), o *GitHub* a plataforma mais utilizada. Os repositórios do *GitHub* apresentam metadadoscomo a quantidade de estrelas, ramos, *commits*e observadores, e a linguagem principal. Esses metadados do *GitHub* permitem quantificar informações sobre o repositório, como o número de pessoas interessadas nele por meio de estrela, observadoresou mesmo seu nível de atividade por meio do número de *commits*. Segundo Borges e Tulio Valente (2018), o número de estrelas é de grande importância como fator de classificação na escolha de repositórios, pois se concluiu que quanto maior o valor desse metadado, maior a probabilidade de pertencer a uma organização e maior o engajamento do número de *commits*e ramos.

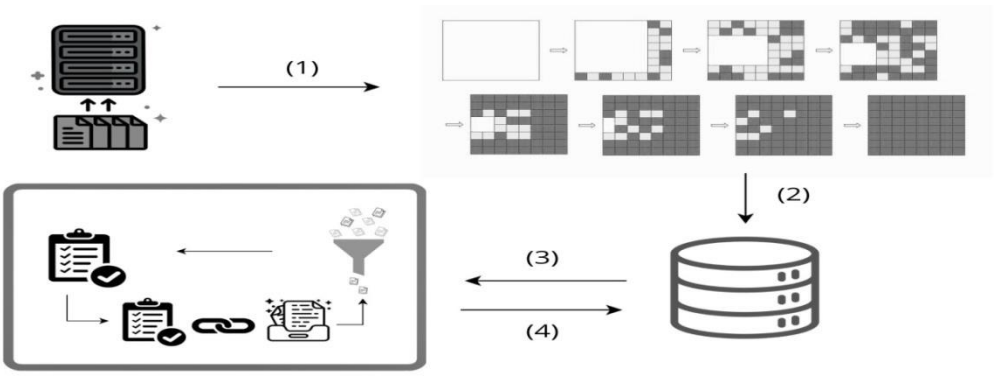
Existem na literatura abordagens que buscam dar suporte ao processo de seleção de repositórios. No entanto, elas abordam o assunto por muitos ângulos. Ferramentas como *GHTorrent*, a busca avançada do *GitHub*, *ModelMine*e *G-Repo* auxiliam o pesquisador dando-lhe acesso a bancos de dados onde é possível extrair os repositórios para estudo. Entretanto, não foram encontradas abordagens que atuam diretamente na documentação dos critérios de seleção de repositórios para auxiliar na reprodutibilidade em experimentos futuros. Essa situação é reforçada pelo estudo de Vidoni (2022),cujo resultado mostra que 83% dos artigos sobre MRS não têm um processo sistemático para escolha de repositórios, enquanto 37% do total nem mesmo mostram o que foi feito para selecionar repositórios para os estudos. A autora destaca que isso pode ser resultado da falta de estruturas ou metodologias para orientar o processo de coleta de informações dos repositórios.

O *RepoDucibility Miner* foi criado para auxiliar pesquisadores no processo de seleção de repositórios e na documentação das decisões tomadas durante as seleções. Com ele, os pesquisadores podem para criar seus próprios conjuntos de dados personalizados para os seus experimentos e serem auxiliados na documentação das decisões tomadas durante a seleção que podem ser armazenas nesta ferramenta. Vale ressaltar que o *RepoDucibility Miner* resolve as limitações identificadas na literatura e, por questão de limite de espaço, recomenda-se a leitura do trabalho anterior (COSTA et al., 2024).

O *RepoDucibility Miner* visa permitir aos pesquisadores definirem os critérios da pesquisa e a plataforma na qual as buscas serão realizadas. Porém, na implementação atual, apenas o *GitHub* foi integrado ao sistema de busca da ferramenta. A Figura 1 apresenta as quatro fases do processo de seleção de repositórios usando o *RepoDucibility Miner*: (Fase 1) a definição de critérios para seleção de repositórios, (Fase 2) a fragmentação das buscas, (Fase 3) a seleção de repositórios e (Fase 4) o enriquecimento dos metadados dos repositórios com dados de pesquisas.

Na Fase 1 é definido o escopo da pesquisa através dos metadados para a seleção de repositórios no *GitHub*, utilizando a *API V4*. É válido ressaltar que a API V4 não permite o filtro pela quantidade de *commits* e outros filtros serão aplicados em fases posteriores com dados enriquecidos obtidos pela abordagem. A Fase (2) consiste na coleta dos repositórios por um processo de fragmentação onde são feitas mini-pesquisas que possibilitam a superação do limite por busca imposto pela *API V4* do *GitHub*. A medida que as mini-pesquisas são feitas, os repositórios retornados são filtrados utilizando os metadados enriquecidos pela abordagem, como a quantidade de *commits*, e armazenados em um banco de dados. A Fase 3 consiste na apresentação desses repositórios e seus metadados para o pesquisador. Nessa fase também é possível criar metadados personalizados — metadados mais subjetivos, como o domínio da aplicação — que ajudam em uma classificação mais objetiva dos dados da pesquisa. Por fim, a Fase 4 consiste no armazenamento do histórico de decisões do pesquisador no banco de dados, enriquecendo os metadados dos repositórios com os dados da pesquisa.

Figura 1 – Abordagem Proposta.



Fonte: (COSTA et al., 2024)

Para exemplificar o funcionamento da ferramenta, foi realizada uma pesquisa por repositórios com linguagem Java e mais de 1.000 estrelas. Como a *API V4* pode filtrar os projetos apenas por linguagem principal, alguns projetos com a linguagem de programação Java seriam ignorados caso esse filtro fosse utilizado. Deste modo, a busca limita-se, em um primeiro momento, apenas em projetos com mais de 1.000 estrelas e retornou 47.448 repositórios em 22 de maio de 2024. Para realizar a pesquisa e superar as limitações da *API*, a busca é fragmentada por número de estrelas, data de criação e tamanho do repositório, permitindo a coleta de até 1.000 repositórios por vez.

O processo de pesquisa e armazenamento demorou aproximadamente 3 horas e 10 minutos e retornou 3.444 repositórios. Para completar a pesquisa para projetos com a linguagem Java, foi recuperada a lista de linguagem de programação em cada um dos 47.448 repositórios, o que resultou em 3.444 repositórios com linguagem Java. É válido mencionar que as pesquisas devem ser autorizadas via *tokens* gerados no GitHub e que neste cenário foi utilizado apenas um *token.* Com o armazenamento dos repositórios, os pesquisadores podem analisar metadados (ex.: número de *commits*, observadores, licenças) e adicionar informações manuais, criando critérios personalizados de análise. Como resultado, tem-se um histórico de análise preciso da seleção dos repositórios junto dos metadados enriquecidos armazenados no banco de dados, tornando os experimentos reprodutíveis, pois os passos são rastreados e a lista de repositórios é persistida.

3 RESULTADOS

Em trabalhos anteriores (COSTA et al., 2024) um experimento de viabilidade foi realizado, considerando três cenários. Todos os cenários consideram repositórios que têm a linguagem de programação *Java* com o número de estrelas sendo maior que 5.000, 10.000 e 50.000 para cada cenário. Como o tempo pode mudar para cada execução, devido a fatores não determinísticos, cada cenário foi executado três vezes. Nesse experimento, apesar de mostrar a viabilidade da ferramenta, mostrou também como o tempo de pesquisa aumenta à medida que se aumenta o número de repositórios pesquisados. Para melhorar a experiência do usuário, foi implementado na ferramenta a possibilidade de se adicionar mais de 1 *token* de pesquisa para tentar diminuir esse tempo de pesquisa.

Para perceber como a quantidade de resultados retornados e a quantidade de *tokens* utilizados impactam na implementação, foram propostas duas questões de pesquisa listadas a seguir:

Questão 1: Qual a influência do número de repositórios retornados no tempo de coleta? Essa questão visa responder como a ferramenta se comporta dado um influxo de variado de informações.

Questão 2: Qual a influência do número de *tokens* no tempo de coleta dos resultados? Essa questão visa responder como a variação no número de *tokens* pode influenciar na velocidade de coleta de dados da ferramenta e se possui um limite máximo na utilidade dos *tokens.*

Para responder às questões abordadas, dois experimentos foram realizados. O primeiro visa responder à Questão 1 realizando pesquisas variando o número de estrelas de 1.000 a 25.000 — com incremento de 2.400 —, utilizando apenas 1 *token*, e coletando a quantidade de repositórios retornados e o tempo de coleta. A Questão 2 foi respondida baseada em um experimento, cujas quantidades de repositórios retornados são, aproximadamente, 5.000, 15.000 e 30.000 repositórios. Deste modo, foram realizadas para repositórios com mais de 8.200, 3.400 e 1.700 estrelas, variando a quantidade de *tokens* de 1 a 5.

Para certificar-se dos valores obtidos dada a natureza não determinística dos resultados, cada experimento foi repetido três vezes para o primeiro experimento e dez vezes para o segundo. Os resultados do primeiro experimento são apresentados Tabela 2, onde se tem os resultados do número de repositórios pesquisados, número de estrelas usados na pesquisa e o tempo médio decorrido delas para 1 *token*. Já o segundo experimento é apresentado na Tabela 3, onde é possível ver o número de estrelas usados na pesquisa, o tempo médio e o desvio padrão para cada *token*.

Tabela 2 – Resultados do primeiro experimento utilizando apenas 1 *token.*

Número de estrelas utilizados na busca	Tempo	Quantidade de repositórios coletados
>25.000	00:04:27	941
>22.600	00:05:08	1.148
>20.200	00:07:03	1.367
>17.800	00:09:01	1.666
>15.400	00:08:29	2.070
>13.000	00:40:50	2.654
>10.600	01:01:45	3.526
>8.200	01:14:23	4.956
>5.800	03:06:21	7.604
>3.400	05:08:25	14.308
>1.000	20:29:10	49.616

Fonte: Autores.

4 DISCUSSÃO

A Tabela 2 apresenta um crescimento do tempo de pesquisa a medida que se aumenta o número de repositórios coletados. Entretanto, essa diferença é mais acentuada em algumas transições. Por exemplo, na transição do intervalo “>15.400” estrelas para o “>13.000” no qual a quantidade de repositórios retornados aumenta de 2.070 para 2.654 (28,21% de crescimento) e o tempo salda de 8:29 minutos para 40:50 (381,34% de crescimento). Outras transições, como de “>10.600” estrelas para “>8.200” e de “>3.400” estrelas para “>1.000”, também apresentam um comportamento acentuado de crescimento. Esse comportamento mostra uma correlação exponencial em relação ao número de repositórios coletados e o tempo de pesquisa. Esse fato foi diagnosticado como resultante do tempo de esperar para poder reutilizar o *token*, quando alcança o limite de buscas,e dos erros secundários gerados pela *API* do *GitHub* que é ativado de maneira inconsistente e várias vezes ao longo da pesquisa.

Tabela 3 – Resultados do segundo experimento utilizando uma variação de *tokens* de 1 a 5 e visualizando a média do tempo e desvio padrão para cada caso de pesquisa.

Resultados		Número de *tokens*				
		1	2	3	4	5
>8.200 estrelas	Média do tempo	01:19:35	00:13:01	00:11:32	00:10:41	00:11:16
	Desvio padrão	00:12:01	00:01:21	00:00:56	00:00:36	00:00:55
>3.400 estrelas	Média do tempo	05:12:15	02:12:03	02:03:05	02:01:14	02:00:05
	Desvio padrão	00:09:46	00:06:02	00:02:50	00:01:31	00:04:26
>1.700 estrelas	Média do tempo	12:11:27	06:06:36	04:09:09	04:03:21	03:59:28
	Desvio padrão	00:12:28	00:05:19	00:05:37	00:04:25	00:01:50

Fonte: Autores.

A Tabela 3 mostra que o aumento do número de *tokens* só traz resultados significativos para até um máximo de 3 *tokens,* sendo possível se perceber uma

redução de 85,51%, 60,58% e 65,94% respectivamente para cada caso apresentado. A partir desse ponto só foi observado um decréscimo pequeno que, no melhor caso, foi de 7,37% quando utilizando 4 *tokens* ao se compara com o uso de 3 *tokens.* Entretanto, para 5 tokens só ouve decréscimo do tempo para o caso que envolvia o maior número de repositórios coletados, o que demonstra que não é necessário utilizar tantos *tokens* para agilizar o processo de pesquisa. Esse fato se deve ao comportamento do *token* de atualizar o número de tentativas, mostrando a capacidade de se circular entre 3 tokens para obter um desempenho de maior custo-benefício. Vale observar que o desvio padrão cai vertiginosamente ao aumentar de 1 para 2 a quantidade de *tokens* e permanecer num declive pequeno até chegar no 5 *token,* onde apenas no caso de mais de 1.700 estrelas apresentou uma redução significativa.

Como resultado, pode-se concluir a Questão 1 que o espaço de pesquisa tende a se comportar de maneira exponencial a medida que se diminui o número de estrelas, o que resulta num tempo maior de pesquisa. Com relação à Questão 2, os *tokens* apesar de serem vitais na melhoria do tempo de pesquisa, acabam sendo realmente valorosos na diminuição do tempo até um máximo de 3 *tokens* por pesquisa. Vale ressaltar também que os dados foram coletados realizando testes em apenas um computador em um ambiente de Internet estável para evitar uma instabilidade maior dos resultados. Outro fator a ser lembrado é que os testes foram feitos apenas para a plataforma do *GItHub*que foi escolhida por ser a mais usada atualmente.

5 CONCLUSÃO

Este artigo apresenta uma abordagem para dar suporte aos estudos de MSR na etapa de seleção de repositórios, rastreando as etapas realizadas pelos pesquisadores. Ele dá suporte à definição dos objetivos da pesquisa, à coleta de grandes quantidades de dados de plataformas que armazenam repositórios e à documentação do racional para selecionar um conjunto de repositórios. É possível observar que o tempo de busca pode crescer em 381,34%% e que o tempo de busca pode reduzir em 85,51% quando 3 *tokens* são utilizados. Porém, ficam claras as limitações dessa coleta dos dados da abordagem por conta de fatores associados à *API V4* do *GitHub.*

Algumas perspectivas para trabalhos futuros são a inserção da ferramenta em um servidor e sua introdução em uma plataforma Web na Internet que permita acesso público; a criação de uma espécie de rede social, possibilitando aos usuários

compartilharem as suas pesquisas e seus dados; e a inserção de novas plataformas de hospedagem de software na plataforma.

6 AGRADECIMENTOS

Os autores agradecem o suporte financeiro provido pela CAPES, CNPq e UFJF.

REFERÊNCIAS

HASSAN, A. E. **The road ahead for Mining Software Repositories**. 2008 Frontiers of Software Maintenance. **Anais**... Em: 2008 IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE. Beijing, China: IEEE, set. 2008.

VIDONI, M. A systematic process for Mining Software Repositories: Results from a systematic literature review. **Information and Software Technology**, v. 144, p. 106791, abr. 2022.

KAGDI, H.; COLLARD, M. L.; MALETIC, J. I. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. **Journal of Software Maintenance and Evolution: Research and Practice**, v. 19, n. 2, p. 77–131, mar. 2007.

BORGES, H.; TULIO VALENTE, M. What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform. **Journal of Systems and Software**, v. 146, p. 112–129, dez. 2018.

REZA, S. M.; BADREDDIN, O.; RAHAD, K. **ModelMine: a tool to facilitate mining models from open source repositories**. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. **Anais**... Em: MODELS '20: ACM/IEEE 23RD INTERNATIONAL CONFERENCE ON MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS. Virtual Event Canada: ACM, 16 out. 2020.

ROMANO, S. et al. **G-Repo: a Tool to Support MSR Studies on GitHub**. 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). **Anais**... Em: 2021 IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, EVOLUTION AND REENGINEERING (SANER). Honolulu, HI, USA: IEEE, mar. 2021.

COSTA, H. H. B. et al. **Tracking the decisions to select repositories for Mining Software Repositories experiments**. Anais Estendidos do XX Simpósio Brasileiro de Sistemas de Informação (SBSI 2024). **Anais**... Em: ANAIS ESTENDIDOS DO SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO. Brasil: Sociedade Brasileira de Computação (SBC), maio 2024.

LUZGIN, V. A.; KHOLOD, I. I. **Overview of Mining Software Repositories**. 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). **Anais**... Em: 2020 IEEE CONFERENCE OF RUSSIAN YOUNG RESEARCHERS IN ELECTRICAL AND ELECTRONIC ENGINEERING (EICONRUS). St. Petersburg and Moscow, Russia: IEEE, jan. 2020.

GÜEMES-PEÑA, D. et al. Emerging topics in mining software repositories: Machine learning in software repositories and datasets. **Progress in Artificial Intelligence**, v. 7, n. 3, p. 237–247, set. 2018.