Desenvolvimento Sustentável de Software e Eficiência Energética do Código
Um Mapeamento Sistemático

André de Moura e Souza, Patrícia Cristiane de Souza, Eunice P. dos Santos Nunes

[1]Instituto de Computação – Universidade Federal de Mato Grosso (UFMT)
CEP 78.060.900 – Cuiabá, MT – Brazil

andre.souza8@sou.ufmt.br, patricia@ic.ufmt.br, eunice.ufmt@gmail.com

Abstract. *Due to the expansion of technological processes, driven by innovations in software models, the importance of sustainability has gained prominence in recent years. This article aims to gather and identify methods and practices that help in the development of software systems aimed at building programs efficiently, aiming at the conservation of computational and environmental resources. In addition, this systematic mapping also addresses the main mechanisms that contribute to the analysis of the application of these techniques.*

Resumo. *Em razão da expansão dos processos tecnológicos, impulsionada pelas inovações nos modelos de software, a importância da sustentabilidade ganhou destaque nos últimos anos. Este artigo tem como objetivo reunir e identificar métodos e práticas que auxiliam no desenvolvimento de sistemas de software voltados para a construção de programas de forma eficiente, visando à conservação de recursos computacionais e ambientais. Além disso, este mapeamento sistemático também aborda os principais mecanismos que contribuem para a análise da aplicação dessas técnicas.*

1. Introdução

O termo "sustentabilidade" originou-se em um contexto em que o uso excessivo dos recursos naturais superava sua capacidade de reposição. Segundo [GROBER 2007], a ideia de sustentabilidade não se restringe apenas ao meio ambiente, mas surge da necessidade de reflexão consolidada na sociedade, sendo peça-chave para mudanças culturais e para o desenvolvimento do pensamento coletivo. Em outras palavras, a sustentabilidade, conforme [FERREIRA 2010], refere-se à condição ou qualidade de algo que pode ser mantido, defendido e sustentado. Trata-se, portanto, de um conceito que vai além das questões ambientais ou econômicas, abrangendo toda a esfera social.

A ideia de desenvolvimento sustentável também passou a estar presente na área de Computação [Rafael 2013]. Nesse contexto, o conceito de TI Verde, definido por Murugesan (2009), abrange o uso da Tecnologia da Informação de forma alinhada aos princípios da sustentabilidade. Inicialmente, antes que o termo ganhasse notoriedade, o objetivo era apenas adotar processos de construção de produtos com foco na agilidade da execução, uma vez que a principal preocupação da TI era a otimização para alcançar a melhor qualidade possível. No entanto, percebeu-se que essa abordagem era insuficiente, pois o uso excessivo de recursos gerava impactos não apenas ambientais, mas também

sociais, comprometendo, inclusive, a própria organização responsável pela produção de TI.

Diante disso, a necessidade de gerenciar os recursos aproximou a Tecnologia da Informação do conceito de sustentabilidade, pois a combinação entre eficiência e uso adequado dos processos contribuiria para a conservação da natureza e, consequentemente, beneficiaria a população que usufrui desses produtos.

Dentro do conceito de TI Verde, a otimização de código para o uso eficiente de energia é um dos pilares fundamentais [WikiC 2022]. A palavra ”otimização”refere-se ao conjunto de melhorias aplicadas para simplificar, ao máximo, os processos envolvidos na solução, sem, contudo, alterar sua funcionalidade. Assim, a otimização ou eficiência do código tem como objetivo modificar algoritmos e programas com a finalidade de reduzir o consumo de recursos, como CPU e memória.

Desta forma, o objetivo deste Mapeamento Sistemático da Literatura (MSL) é identificar processos que contribuam para a construção de produtos alinhados ao conceito de TI Verde, com foco específico no desenvolvimento de software e na eficiência do código. Em outras palavras, este artigo busca reunir técnicas que auxiliem na elaboração de algoritmos otimizados e eficientes, integrando, simultaneamente, os princípios da sustentabilidade.

2. Metodologia de Pesquisa

A metodologia utilizada na execução do mapeamento sistemático segue as orientações de Kitchenham e Charters [Kitchenham 2007]. A Figura 1 demonstra esse método.

As questões de pesquisa foram elaboradas para abranger todo o escopo do desenvolvimento sustentável de software e a eficiência energética do código. Neste contexto, as questões de pesquisa deste mapeamento são as seguintes:

Q1: Quais as práticas e técnicas utilizadas para a elaboração de algoritmos otimizados?

Q2: Quais os processos usados na verificação da eficiência do processamento do código?

Q3: Quais os benefícios e as vantagens, em geral, na aplicação das técnicas apresentadas na Q1?

As bibliotecas digitais utilizadas como fonte de busca foram a IEEE Xplore, ACM Digital Library e Google Acadêmico (manual). A IEEE e a ACM são umas das principais bases que agregam artigos relacionados à área de Ciência da Computação. O Google Acadêmico reúne diversos bancos de dados bibliográficos em importantes periódicos e conferências.

Todos os artigos encontrados, a partir da execução da string de busca, foram avaliados e aplicados os passos conforme os critérios de inclusão e exclusão estabelecidos no protocolo. Os seguintes critérios de inclusão considerados foram: (i) Os artigos devem estar escritos em português ou inglês; (ii) Os artigos devem conter as palavras chaves utilizadas na string de busca no título e/ou no resumo e/ou nas palavras-chave; (iii) Os trabalhos devem abordar os processos algorítmicos; (iv) Considerados apenas artigos publicados entre 2019 a 2025, esse período foi definido pela razão do tema ganhar mais

destaque recentemente, por isso nesse intervalo de tempo apresenta artigos mais relevantes sobre essa temática. A negação desses requisitos fazem parte dos critérios de exclusão, além de outros itens, como (i) Remoção de artigos duplicados; (ii) Excluir Artigos que não possuem acesso aberto; (iii) Desconsiderar artigos que não são de natureza primária.

A string utilizada para a busca nas bibliotecas foi a seguinte:

Tabela 1. String De Busca

Idioma	String de Busca
Inglês	("sustainable software development" OR "green software engineering" OR "energy-aware software") AND ("energy efficiency" OR "low-power computing" OR "power consumption" OR "energy-aware programming") AND ("code optimization" OR "algorithm efficiency" OR "software performance" OR "resource-efficient coding")

A figura 1 mostra o Processo do Mapeamento Sistemático:

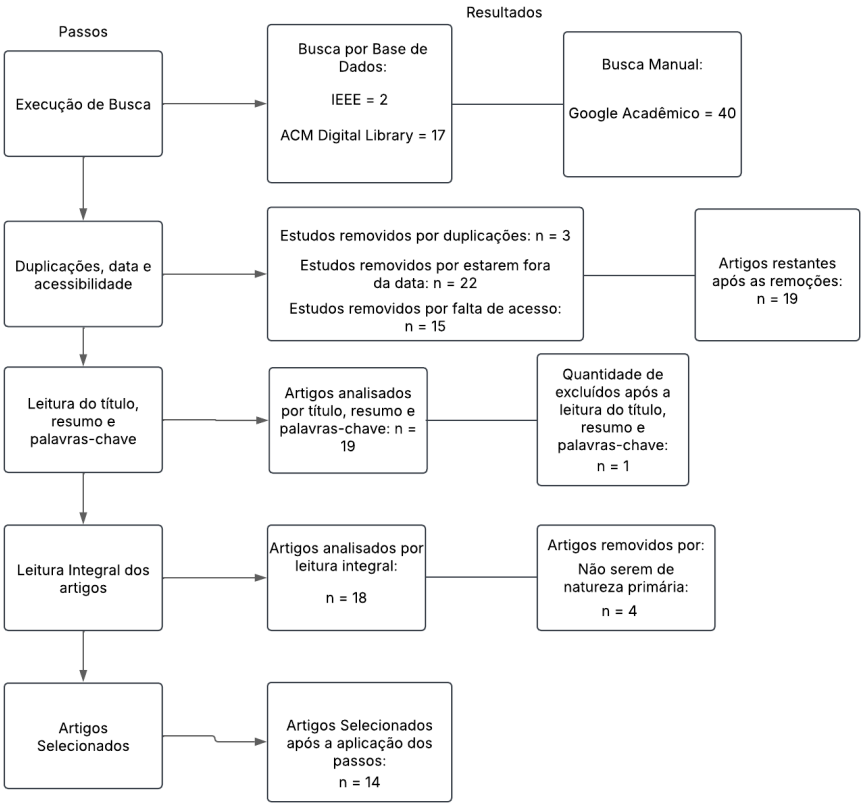

Figura 1. Processo do Mapeamento Sistemático

Observa-se que foram identificadas 59 pesquisas no total, considerando tanto a busca em bases de dados quanto a busca manual. Inicialmente, foi realizada uma filtragem para eliminar artigos duplicados, publicados fora do período definido no protocolo ou que não possuíam acesso aberto, resultando na exclusão de 40 deles. Em seguida, foi

feita uma leitura preliminar dos títulos, resumos e palavras-chave dos artigos restantes, etapa na qual apenas 1 artigo foi removido. Posteriormente, procedeu-se à leitura integral dos textos restantes, o que levou à exclusão de mais 4 pesquisas por não se tratarem de estudos primários. Assim, restaram 14 artigos, que compõem as discussões apresentadas na próxima seção.

É importante ressaltar que, ao realizar a busca no Google Acadêmico utilizando a string definida, foram retornados 271 resultados. No entanto, no processo manual de triagem, apenas os 40 primeiros artigos foram considerados para avaliação.

3. Discussão dos Resultados

Os artigos selecionados após a aplicação dos critérios foram submetidos a um estudo com o objetivo de avaliar e analisar as técnicas e os procedimentos utilizados no desenvolvimento de softwares sustentáveis.

A Tabela 2 apresenta um resumo dos resultados obtidos por meio da análise dos 14 artigos selecionados no Mapeamento Sistemático da Literatura, ordenados por data de publicação. As principais informações contidas na tabela são: as fontes, as técnicas utilizadas para a otimização do código (Q1) e os processos empregados na verificação dos resultados (Q2).

Com base nos dados extraídos da tabela, é possível analisar algumas das técnicas empregadas na otimização de código, que contribuem para a construção de softwares mais eficientes e sustentáveis. Uma das técnicas identificadas foi a Power-Optimised Software Envelope (POSE) [Roberts et al. 2019], uma ferramenta que auxilia na análise do desempenho de sistemas de software, considerando tanto o tempo de execução quanto o consumo de energia.

Práticas de melhoria de algoritmos são amplamente utilizadas, como a refatoração de código [Connolly and Ó Cinnéide 2024] [Yadav et al. 2024] [Kile et al. 2025], que permite a simplificação de blocos de código, além de facilitar a modularização e a padronização. Esses processos não só garantem a eficiência, mas também melhoram a legibilidade do algoritmo. Tal melhoria pode ser testada utilizando modelos que simulam o desempenho do software, possibilitando uma avaliação mais precisa. Além disso, é importante enfatizar que a escolha de trechos de código deve ser analisada com base no contexto, pois, em determinadas situações, um algoritmo pode ser mais adequado a um cenário específico. Um exemplo disso são os algoritmos de ordenação: Bubble Sort e Quick Sort [Felix and Mohankumar 2024].

Em Python [Reya et al. 2023], a utilização de estruturas de dados eficazes, o uso de técnicas de programação dinâmica e a escolha adequada de operações estão entre as práticas comuns para a otimização de código. Esses processos foram amplamente empregados na melhoria de sistemas voltados à energia eólica [Robinson 2024].

Destaca-se, também, o uso do Energy-Aware Prompting (EAP) [Peng et al. 2024], uma ferramenta desenvolvida com o objetivo de elaborar prompts otimizados para auxiliar na criação de códigos com foco em eficiência energética.

Em geral, o uso de métricas de desempenho, a construção eficiente de algoritmos, a avaliação de código, a aplicação de técnicas de Machine Learning [Biswas 2023], o teste de comandos e variáveis, a comparação da execução de tarefas, a análise do con-

sumo energético potencial e a implementação estratégica de arquiteturas de hardware são práticas comuns que contribuem para o desenvolvimento sustentável de software e para a mitigação do consumo energético.

Também é possível extrair, a partir da tabela, informações relevantes sobre o uso de determinadas ferramentas e técnicas para a verificação do processamento de algoritmos, uma etapa imprescindível na avaliação do consumo de energia. Um dos modelos criados para medir o comportamento energético durante a execução do código foi o Energy Labelled Transitions System (ELTS) [Ferreira 2022]. Esse modelo permite a definição das Basic Energy Units (BETs), que servem para associar trechos relevantes do código a elementos específicos do ELTS, possibilitando uma avaliação detalhada do consumo energético por meio do rastreamento entre o código e o modelo.

O uso de hardware para medir o consumo de energia também pode ser aplicado, como no caso do Power Reading United (PRU) [Felix and Mohankumar 2024], utilizado como um componente integrado ao computador. Esse dispositivo auxiliou no monitoramento do consumo energético de programas construídos com os algoritmos Bubble Sort e Quick Sort, de forma controlada, contínua e permitindo comparações durante a execução.

A utilização de bibliotecas, como a CodeCarbon para Python [Reya et al. 2023], também contribui para o monitoramento do consumo de energia. Esse tipo de ferramenta, baseado na importação de bibliotecas específicas, é conhecido como Energy Profiling [Meinhardt 2024].

Destaca-se, também, o uso de Benchmarks [Rinne 2024], utilizados não apenas para medir o consumo de energia, mas também para monitorar a sustentabilidade ambiental por meio da análise do uso de recursos durante a execução do código.

O uso de softwares, como o ”Performance Profiler”, integrante de uma ferramenta da Microsoft, auxiliou na análise do desempenho do código à medida que verificava as condições da CPU e da memória em ambientes de computação em nuvem [Alsayyah and Ahmed 2020].

Em geral, o monitoramento do uso de energia e do tempo de execução, o acompanhamento do desempenho dos processos do programa e da CPU, a utilização de ferramentas e softwares como auxiliares na medição do consumo, a realização de testes repetitivos nos códigos, bem como a comparação e a análise dos resultados, são práticas comuns na verificação do processamento do código e do seu consumo energético.

O uso de ferramentas e a adoção de práticas voltadas à otimização e ao desenvolvimento sustentável de software oferecem diversas vantagens, tais como: redução significativa do consumo de energia, possibilitando um processamento mais rápido e eficiente; diminuição de custos operacionais; contribuição para a computação verde; padronização de práticas de codificação sustentável; melhoria no desempenho do software e no gerenciamento de recursos; prolongamento da vida útil dos equipamentos; aprimoramento da experiência do usuário; e redução da pegada de carbono.

As ferramentas listadas até aqui são de grande utilidade para a construção de códigos otimizados. No entanto, algumas dessas técnicas não foram testadas em todas as situações possíveis. Um exemplo disso é a refatoração de código, apresentada em um dos artigos, que foi utilizada em uma situação específica para comparar os algoritmos Bubble

Tabela 2. Principais informações extraídas dos artigos

Fonte	técnicas usados	Processos de verificação
[Roberts et al. 2019]	Power-Optimised Software Envelope (POSE)	Medição de consumo de energia
[Alsayyah and Ahmed 2020]	fase de desenvolvimento parametrizada	Uso do software ”performance profile”
[Ferreira 2022]	Substituição de estruturas de dados	Energy Labelled Transitions System (ELTS)
[Biswas 2023]	Aplicação de frameworks e Machine Learning	Execução de testes e análise dos resultados
[Reya et al. 2023]	Aplicação de estruturas de dados eficientes	Uso de bibliotecas para monitorar o consumo de energia
[Connolly and Ó Cinnéide 2024]	Refatoração de Código	Cálculo de métricas ponderadas
[Rinne 2024]	Otimização de estruturas de dados	Uso de Benchmarks
[Felix and Mohankumar 2024]	Utilização de algoritmos adequados	Uso da Power Reading United (PRU)
[Kandimalla and Bolla 2024]	Utilização de algoritmos adequados	Monitoramento da CPU e consumo de energia
[Meinhardt 2024]	Otimização dos designs dos algoritmos	Uso de Energy Profiling
[Peng et al. 2024]	Uso de Energy-Aware Prompting (EAP)	Medição do consumo de energia
[Robinson 2024]	Projeção de design dos algoritmos	Monitoramento do desempenho do programa
[Yadav et al. 2024]	Refatoração de Código	Análise do tempo de execução e consumo de energia
[Kile et al. 2025]	Refatoração de código	Avaliação do tempo de execução e uso de CPU

Sort e Quick Sort. Por esse motivo, não é possível afirmar que essas técnicas serão sempre eficientes quando aplicadas, podendo haver limitações ou a revelação de problemas nas ferramentas, o que exige um estudo mais aprofundado sobre essas técnicas.

É importante lembrar que a preocupação com a construção de algoritmos otimizados e sustentáveis ainda é um desafio. Como a ideia ganhou mais destaque recentemente, algumas empresas que decidiram adotar o conceito em seu processo de produção lidam com a situação de desenvolvedores que enfrentam dificuldades na aplicação dessas técnicas [Kandimalla and Bolla 2024]. Adotar novas medidas requer mudanças no padrão de desenvolvimento de software e esforço da equipe de TI para se habituar a esse novo modelo. Contudo, isso provavelmente seja apenas uma questão de tempo até que a maioria das empresas consiga adaptar-se a essa nova realidade.

4. Considerações finais

Em suma, a aplicação de técnicas de otimização de código sempre dependerá de certos contextos e fatores, pois não existem práticas uniformes que sirvam para todas as situações. As questões de eficiência na execução e consumo de energia são duas peças-chave que norteiam o desenvolvimento de sistemas de software, uma vez que, com essas ideias em mente, é possível encontrar o equilíbrio entre a rapidez dos processos e o gasto de recursos materiais e energéticos.

Este mapeamento sistemático da literatura contribuiu para a listagem e identificação de práticas que colaboram na construção de algoritmos otimizados, sem, no entanto, aprofundar no funcionamento de cada uma delas. Cada técnica apresenta variações em termos de rendimento e eficiência; contudo, não é possível determinar quais métodos são mais utilizados, nem a total eficiência e desempenho dessas técnicas em todas as suas aplicações.

É importante salientar que o estudo da sustentabilidade na área de computação ganhou destaque mais recentemente. Portanto, ainda é possível que surjam novas descobertas e conceitos sobre otimização de programas e sistemas de software no futuro.

Sendo assim, o mapeamento sistemático da literatura identificou diversas ferramentas que auxiliam no desenvolvimento sustentável de software. Como trabalhos futuros, pretende-se dar continuidade ao estudo dessas técnicas, realizando diversos experimentos para verificar suas limitações e aprimorá-las. Além disso, planeja-se realizar uma pesquisa comparativa entre essas ferramentas, no contexto de sustentabilidade de sistemas.

5. Agradecimentos

O NotebookLM foi utilizado para auxiliar na leitura e na triagem dos artigos selecionados e o ChatGPT para a análise e modificações da string de busca.

Referências

Alsayyah, A. A. and Ahmed, S. (2020). Energy efficient software development techniques for cloud based applications. *International Journal*, 9(5).

Biswas, M. P. (2023). Estimating the energy cost of scientific software.

Connolly, D. and Ó Cinnéide, M. (2024). Weighted metrics for the development of energy efficient software. In *Proceedings of the 1st International Workshop on Designing Software*, Designing '24, page 64–69, New York, NY, USA. Association for Computing Machinery.

Felix, P. and Mohankumar, M. (2024). Energy optimization in software development: A comparative study of sorting techniques.

FERREIRA, A. B. (2010). *Dicionário Aurélio da Língua Portuguesa*. Curitiba, Paraná: Positivo - Livros, 5 edition.

Ferreira, O. A. (2022). Modelling software energy consumption for energy efficiency analysis.

GROBER, U. (2007). *Deep Roots: A Conceptual History of "sustainable Development" (Nachhaltigkeit)*. Discussion papers, Wissenschaftszentrum Berlin für Sozialforschung. Berlin: WZB.

Kandimalla, A. M. and Bolla, A. (2024). Green coding practices in software development: Perceptions, barriers, and strategies for change.

Kile, S. A., Iliyas, I. I., and Bassi, J. Y. (2025). Energy efficient software development: An integrated approach for green computing and enhanced software performance. *BIMA JOURNAL OF SCIENCE AND TECHNOLOGY (2536-6041)*, 8(4B):212–224.

Kitchenham, B. Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering*. Software Engineering Group Department of Computer Science Keele University.

Meinhardt, M. (2024). Energy-efficiency tactics in software architecture and implementation.

Peng, H., Gupte, A., Eliopoulos, N. J., Ho, C. C., Mantri, R., Deng, L., Jiang, W., Lu, Y.-H., Läufer, K., Thiruvathukal, G. K., et al. (2024). Large language models for energy-efficient code: Emerging results and future directions. *arXiv preprint arXiv:2410.09241*.

Rafael (2013). Ti sustentável: conceito, soluções e consequências. Disponível em: https://www.devmedia.com.br/ti-sustentavel-conceito-solucoes-e-consequencias/29394.

Reya, N. F., Ahmed, A., Zaman, T., and Islam, M. M. (2023). Greenpy: evaluating application-level energy efficiency in python for green computing. *Annals of Emerging Technologies in Computing (AETiC)*, 7(3):92–110.

Rinne, T. (2024). Adapting sustainable software development methods into agile processes.

Roberts, S. I., Wright, S. A., Fahmy, S. A., and Jarvis, S. A. (2019). The power-optimised software envelope. *ACM Trans. Archit. Code Optim.*, 16(3).

Robinson, T. (2024). Sustainable development through green software engineering: An empirical study in wind energy.

WikiC (2022). Wiki computação. Disponível em: https://wiki.inf.ufpr.br/computacao/doku.php?id=o:otimizacao.

Yadav, A., Usman, M., Sati, A., and Jain, S. (2024). Revolutionizing software development: Enhancing quality and performance through code refactoring. In *Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing*, IC3-2024, page 715–725, New York, NY, USA. Association for Computing Machinery.