Um Framework para Publicação Reprodutível em Ciência Computacional: Dos Artefatos ao Impacto

Seção 1: Estruturas Fundamentais para a Pesquisa Reprodutível

A ascensão da ciência computacional transformou a própria natureza da investigação científica. Onde antes o caderno de laboratório era o registro imutável do processo experimental, hoje o código-fonte, os conjuntos de dados e os fluxos de trabalho computacionais formam a base da descoberta. No entanto, essa transição introduziu um novo desafio: a "crise de reprodutibilidade", na qual estudos, apesar de publicados, não podem ser independentemente verificados devido à falta de acesso aos materiais subjacentes ou à documentação inadequada.[1] A reprodutibilidade computacional, definida como a capacidade de usar os materiais de um estudo anterior (como dados, código e documentação) para regenerar os resultados, incluindo figuras e tabelas, não é meramente uma questão de rigor técnico; é um pilar fundamental do método científico na era digital.[1] A reprodutibilidade é o primeiro passo em direção à confiabilidade geral de um estudo. Se um estudo é reprodutível, quaisquer problemas analíticos que possam invalidar os resultados podem ser mais facilmente identificados e corrigidos, fortalecendo a estrutura da ciência cumulativa.[1] Este relatório estabelece um framework abrangente para garantir que os artefatos de um projeto de pesquisa — código, dados e experimentos — não apenas atendam, mas excedam os padrões de reprodutibilidade e reuso exigidos para publicação científica de alto impacto.

1.1 Sintetizando os Pilares e as Regras: Uma Doutrina Unificada

Para construir um projeto computacional robusto e reprodutível, é necessário um alicerce de princípios orientadores. Duas das estruturas mais influentes neste domínio são os "Cinco

Pilares da Reprodutibilidade Computacional" e as "Dez Regras Simples para Pesquisa Computacional Reprodutível". Em vez de serem vistos como abordagens concorrentes, esses frameworks são complementares, oferecendo uma visão em múltiplas camadas, desde a estratégia de alto nível até a tática de implementação no nível do solo.

Os Cinco Pilares fornecem a arquitetura estrutural de um projeto reprodutível. Eles são [1]:

1. **Programação Literária (Literate Programming):** A fusão de código, texto e resultados em um único documento dinâmico, garantindo que a narrativa e a análise estejam intrinsecamente ligadas.
2. **Controle de Versão e Compartilhamento de Código (Code Version Control and Sharing):** O rastreamento sistemático de todas as alterações no código e em outros artefatos baseados em texto, facilitando a colaboração e a auditoria histórica.
3. **Controle do Ambiente Computacional (Compute Environment Control):** A captura e o arquivamento precisos de todas as dependências de software, desde a versão do sistema operacional até as bibliotecas de pacotes, para garantir que o código seja executado de forma idêntica no futuro.
4. **Compartilhamento Persistente de Dados (Persistent Data Sharing):** A prática de tornar os dados Findable, Accessible, Interoperable, and Reusable (FAIR), garantindo sua disponibilidade a longo prazo.
5. **Documentação (Documentation):** A criação de documentação clara e abrangente que descreve não apenas *como* o projeto funciona, mas *por que* foi projetado de tal maneira.

Enquanto os Pilares definem o "o quê", as Dez Regras Simples fornecem o "como", oferecendo táticas acionáveis que implementam diretamente esses princípios.[7] Por exemplo, a

Regra 1: Para Cada Resultado, Rastreie Como Foi Produzido e a **Regra 2: Evite Passos Manuais de Manipulação de Dados** são implementações diretas do pilar da Programação Literária e da automação de fluxo de trabalho. Elas abordam a fragilidade da documentação manual, que pode facilmente ficar dessincronizada com a análise real, e promovem a criação de fluxos de trabalho executáveis.[7] Da mesma forma, a

Regra 3: Arquive as Versões Exatas de Todos os Programas Externos Usados e a **Regra 4: Controle a Versão de Todos os Scripts Personalizados** são a aplicação prática dos pilares de Controle do Ambiente Computacional e Controle de Versão de Código, respectivamente.[7]

A convergência desses frameworks revela um princípio central que transcende a mera documentação: o princípio da **Proveniência Executável**. A pesquisa computacional oferece uma mudança de paradigma em relação à ciência experimental tradicional. Em um laboratório úmido, o caderno de anotações é um registro passivo do que *foi feito*. Na ciência computacional, a documentação pode se tornar o próprio experimento. Um script, um notebook de programação literária ou um Makefile não é apenas uma descrição do fluxo de

trabalho; é uma especificação precisa, inequívoca e, crucialmente, *executável* de toda a análise, desde os dados brutos até a figura final.[7] Este conceito de proveniência executável elimina a possibilidade de divergência entre o que é relatado e o que foi realmente executado, uma das fontes mais comuns de irreprodutibilidade.[7] Portanto, o objetivo final de um projeto reprodutível não é apenas criar artefatos bem documentados, mas construir um sistema integrado onde um único comando (por exemplo,

make all) pode regenerar todos os resultados, tabelas e figuras a partir dos dados brutos e do código-fonte. Este se torna o teste definitivo de reprodutibilidade.

1.2 O Espectro da Reprodutibilidade: Da Verificação à Generalização

É fundamental estabelecer uma terminologia clara para discutir os diferentes níveis de validação científica que a reprodutibilidade permite. O espectro da reprodutibilidade pode ser definido da seguinte forma [1]:

- **Reprodutibilidade (Reproducibility):** Refere-se à capacidade de obter os mesmos resultados computacionais usando o *mesmo código* e os *mesmos dados* do estudo original. Este é o nível mais fundamental de verificação e o foco principal deste relatório.
- **Replicabilidade (Replicability):** Refere-se à capacidade de obter resultados consistentes ao realizar um *novo estudo* que aborda a mesma questão científica. Isso geralmente envolve a coleta de novos dados e, potencialmente, a implementação de um novo código para testar a mesma hipótese.
- **Robustez (Robustness):** Refere-se à capacidade de obter resultados consistentes ao fazer pequenas perturbações na análise, como usar um conjunto de dados ligeiramente diferente ou alterar os parâmetros do modelo.

Este relatório se concentra em alcançar o que pode ser chamado de "reprodutibilidade extrema" [5], que é a base sobre a qual todas as outras formas de validação são construídas. Um projeto que é perfeitamente reprodutível permite que outros pesquisadores não apenas verifiquem os resultados declarados, mas também testem a replicabilidade e a robustez das descobertas, investigando o comportamento do método em novas condições e com novos dados. Sem essa base sólida de reprodutibilidade, a ciência cumulativa é paralisada.

Seção 2: Implementando um Fluxo de Trabalho Reprodutível: Uma Pilha de Tecnologia

A transição dos princípios abstratos de reprodutibilidade para a prática concreta exige a adoção de uma pilha de tecnologia moderna e bem integrada. Esta seção detalha as ferramentas essenciais para implementar um fluxo de trabalho reprodutível, fornecendo uma análise comparativa profunda para orientar as escolhas de tecnologia com base na complexidade e nas necessidades do projeto.

2.1 Controle de Versão de Código e Ativos: Git, Git LFS e DVC

O controle de versão é a espinha dorsal de qualquer projeto de software sério e, por extensão, de qualquer projeto de pesquisa computacional. Ele atende diretamente ao pilar de "Controle de Versão e Compartilhamento de Código" e à "Regra 4: Controle a Versão de Todos os Scripts Personalizados".[3]

A Base: Git
O Git é o padrão de fato para o controle de versão distribuído. Sua função é rastrear todas as alterações feitas em arquivos baseados em texto (código-fonte, scripts, documentação, arquivos de configuração). Os benefícios do Git são múltiplos e fundamentais: ele mantém um histórico completo de todas as alterações, permitindo a inspeção e execução do código em qualquer ponto passado; facilita a colaboração entre membros da equipe; e protege contra a perda de código.3 Para todos os artefatos de texto em um projeto de pesquisa, o uso do Git não é negociável.
O Problema dos Arquivos Grandes: Git LFS
O Git foi projetado para arquivos de texto e seu desempenho se degrada significativamente ao tentar versionar arquivos binários grandes, como conjuntos de dados, imagens de contêineres ou modelos de aprendizado de máquina treinados. A tentativa de fazer isso resulta em repositórios inchados e operações de clone e pull lentas. A solução padrão para esse problema é o Git Large File Storage (Git LFS). O Git LFS funciona substituindo arquivos grandes no repositório Git por pequenos arquivos de ponteiro baseados em texto. Esses ponteiros contêm metadados, como um hash do conteúdo do arquivo, enquanto o arquivo real é armazenado em um servidor LFS remoto. Esse processo é transparente para o usuário; comandos Git padrão como git add, git push e git pull funcionam como de costume, com o cliente LFS interceptando e gerenciando as transferências de arquivos grandes em segundo plano.3
Uma Alternativa Centrada em Dados: DVC
Embora o Git LFS resolva o problema de armazenamento de arquivos grandes, ele tem uma limitação fundamental para fluxos de trabalho de aprendizado de máquina: ele é "inconsciente do fluxo de trabalho". O Git LFS versiona arquivos individuais, mas não tem conhecimento das relações de dependência entre eles. Ele não sabe que um modelo

específico (model.pkl) foi treinado usando uma versão particular de um script (train.py) e um conjunto de dados (data.csv). Essa lacuna de proveniência é onde o Data Version Control (DVC) se destaca.10

O DVC é uma ferramenta de código aberto projetada especificamente para o ciclo de vida do aprendizado de máquina.[13] Ele opera em conjunto com o Git, mas estende suas capacidades para gerenciar não apenas dados e modelos, mas também as pipelines que os conectam. As principais vantagens do DVC são:

- **Consciência da Pipeline:** O DVC permite definir uma pipeline de múltiplos estágios em um arquivo dvc.yaml. Cada estágio define suas dependências (entradas) e saídas. Ao executar dvc repro, o DVC verifica quais dependências mudaram e reexecuta apenas os estágios afetados da pipeline, garantindo a reprodutibilidade e economizando tempo computacional.[8]
- **Rastreamento de Experimentos:** O DVC pode rastrear experimentos, registrando métricas e parâmetros juntamente com as versões de código e dados, permitindo comparações detalhadas entre diferentes execuções.[8]
- **Flexibilidade de Armazenamento:** Ao contrário do Git LFS, que normalmente está vinculado ao armazenamento do provedor Git (por exemplo, GitHub, GitLab), o DVC é agnóstico em relação ao armazenamento e suporta uma ampla gama de back-ends, incluindo Amazon S3, Google Cloud Storage, Azure Blob Storage, SSH ou mesmo diretórios locais.[8]

A escolha entre Git LFS e DVC depende criticamente da complexidade do fluxo de trabalho do projeto. Para projetos com ativos grandes e estáticos — por exemplo, um único conjunto de dados que é analisado de várias maneiras, mas nunca modificado — a simplicidade e a integração transparente do Git LFS podem ser suficientes. No entanto, a maioria dos projetos de aprendizado de máquina não é estática; eles são Grafos Acíclicos Dirigidos (DAGs) de dependências, onde os dados brutos são processados, os recursos são projetados, os modelos são treinados e as métricas são avaliadas. Para esses projetos, a incapacidade do Git LFS de rastrear essas dependências é uma falha crítica para a reprodutibilidade. O DVC, sendo "consciente do DAG", foi projetado para resolver precisamente este problema. Portanto, existe um "limiar de complexidade do fluxo de trabalho". Abaixo desse limiar, o Git LFS é uma solução viável. Acima dele, o custo de aprendizado adicional do DVC é amplamente justificado pelos ganhos maciços em automação, proveniência e reprodutibilidade garantida. Para qualquer projeto que envolva mais do que uma única etapa de processamento de dados ou treinamento de modelo, o DVC é a escolha superior para garantir a reprodutibilidade científica.

Característica	Git LFS	DVC (Data Version Control)
Caso de Uso Principal	Gerenciamento transparente de arquivos	Versionamento de dados e modelos, gerenciamento

	binários grandes dentro de um fluxo de trabalho Git padrão.	de pipelines de ML e rastreamento de experimentos.
Granularidade de Versionamento	Nível de arquivo. Rastreia o conteúdo de arquivos individuais.	Nível de arquivo e pipeline. Rastreia arquivos e as dependências entre código, dados e saídas.
Consciência da Pipeline	Nenhuma. Não tem conhecimento das relações entre código, dados e modelos.	Fundamental. Define e versiona pipelines de múltiplos estágios em dvc.yaml, permitindo a reprodução com dvc repro.
Rastreamento de Experimentos	Nenhum. Não rastreia métricas ou hiperparâmetros.	Integrado. Pode rastrear e comparar métricas, parâmetros e gráficos entre experimentos.
Backend de Armazenamento	Geralmente acoplado ao provedor Git (ex: GitHub, GitLab), com custos associados.	Flexível e agnóstico. Suporta armazenamento em nuvem (S3, GCS, Azure), SSH, HDFS ou armazenamento local.
Facilidade de Uso	Curva de aprendizado baixa. Integra-se de forma transparente com os comandos Git existentes.	Curva de aprendizado mais alta. Requer o aprendizado de comandos DVC além do Git.
Ecossistema/Comunidade	Amplamente suportado por plataformas Git e ferramentas de CI/CD.	Comunidade de código aberto ativa e crescente, focada especificamente em MLOps e reprodutibilidade.

2.2 Gerenciamento de Ambiente e Dependências: Conda vs. Poetry

O pilar de "Controle do Ambiente Computacional" e a "Regra 3: Arquive as Versões Exatas de Todos os Programas Externos Usados" destacam uma verdade crítica: o código não existe no vácuo.[1] Ele é executado dentro de um ambiente computacional complexo, e variações nesse ambiente podem levar a resultados diferentes ou a falhas completas. Gerenciar esse ambiente de forma explícita e reprodutível é, portanto, essencial.

Análise Comparativa das Ferramentas

Duas ferramentas proeminentes para gerenciamento de ambiente e dependências no ecossistema Python são Conda e Poetry.

- **Conda:** É um gerenciador de pacotes, dependências e ambientes de código aberto e multiplataforma. Sua principal força reside na capacidade de gerenciar não apenas pacotes Python, mas também bibliotecas e binários de outras linguagens (por exemplo, C/C++, R) e dependências complexas como o CUDA para computação em GPU. Isso o torna extremamente popular na comunidade de ciência de dados, onde pacotes científicos como NumPy, SciPy e PyTorch dependem fortemente de bibliotecas compiladas subjacentes.[18] O Conda se destaca na criação de ambientes isolados que encapsulam todas essas dependências.
- **Poetry:** É uma ferramenta moderna para gerenciamento de dependências e empacotamento em Python. Ele adere aos padrões modernos de empacotamento Python (PEP 517/518) usando um único arquivo pyproject.toml para gerenciar metadados do projeto e dependências. A principal vantagem do Poetry para a reprodutibilidade é seu mecanismo de resolução de dependências determinístico e a criação de um arquivo poetry.lock. Este arquivo de bloqueio registra as versões exatas de cada pacote e de suas dependências transitivas, garantindo que o mesmo ambiente possa ser recriado de forma idêntica em diferentes máquinas ou em momentos diferentes.[18]

A Estratégia de Ambiente Híbrido

Embora muitas vezes apresentados como alternativas, Conda e Poetry resolvem problemas ligeiramente diferentes e podem ser usados de forma complementar para criar um ambiente de pesquisa idealmente robusto e reprodutível. A computação científica frequentemente depende de pacotes com dependências binárias complexas, que o Conda gerencia de forma soberba. No entanto, a reprodutibilidade rigorosa exige um bloqueio determinístico de dependências, que é a principal força do Poetry.

Uma estratégia híbrida aproveita o melhor de ambos os mundos:

1. **Use o Conda para o Nível do Ambiente:** Crie um ambiente Conda para instalar o interpretador Python de uma versão específica e quaisquer dependências não-Python ou binárias pesadas (por exemplo, cudatoolkit, cudnn). Esta etapa estabelece a base estável do ambiente.
2. **Use o Poetry para o Nível do Pacote Python:** Dentro do ambiente Conda ativado, use o Poetry para gerenciar todas as dependências de pacotes Python. O Poetry irá resolver a árvore de dependências, gerar o arquivo poetry.lock determinístico e instalar os

pacotes.

Esta abordagem híbrida garante que as complexas dependências de nível de sistema sejam tratadas pelo Conda, enquanto a reprodutibilidade exata do ambiente Python é garantida pelo Poetry. Para publicar ou compartilhar o projeto, ambos os arquivos de configuração (environment.yml para Conda e pyproject.toml/poetry.lock para Poetry) devem ser incluídos no controle de versão.

Característica	Conda	Poetry
Escopo Principal	Gerenciador de pacotes e ambientes para múltiplas linguagens (Python, R, C++, etc.).	Gerenciador de dependências e empacotamento focado em Python.
Resolução de Dependências	Robusta, mas pode ser lenta e nem sempre determinística por padrão.	Rápida e determinística. Gera um arquivo poetry.lock para compilações reprodutíveis.
Isolamento de Ambiente	Capacidade nativa principal. Cria ambientes totalmente isolados.	Utiliza ambientes virtuais Python padrão (venv), que pode gerenciar internamente.
Fontes de Pacotes	Canais Conda (ex: anaconda, conda-forge), que hospedam pacotes pré-compilados.	PyPI (Python Package Index) e outros repositórios de pacotes Python.
Manipulação de Dependências Não-Python	Ponto forte. Gerencia facilmente bibliotecas C/C++, CUDA, etc.	Limitado. Focado principalmente em pacotes Python e suas dependências.
Garantia de Reprodutibilidade	Boa, com environment.yml. Aprimorada com ferramentas como conda-lock.	Excelente. O arquivo poetry.lock garante a recriação bit a bit do ambiente de pacotes.

2.3 Automação de Ponta a Ponta: O Poder dos Makefiles

A automação é a pedra angular da reprodutibilidade. A "Regra 2: Evite Passos Manuais de Manipulação de Dados" é um mandato para substituir procedimentos manuais, que são ineficientes e propensos a erros, por fluxos de trabalho programáticos.[7] A ferramenta

make, com seu arquivo de configuração Makefile, é uma solução clássica e poderosa para automatizar fluxos de trabalho de pesquisa.[23]

Um Makefile é estruturado em torno de três conceitos fundamentais [23]:

- **Alvos (Targets):** Os arquivos que se deseja criar (por exemplo, report.pdf, results.csv).
- **Pré-requisitos (Prerequisites):** Os arquivos dos quais um alvo depende (por exemplo, report.pdf depende de analysis.R e data.csv).
- **Receitas (Recipes):** Os comandos a serem executados para criar o alvo a partir de seus pré-requisitos.

O make constrói um grafo de dependências a partir dessas regras. Quando invocado, ele verifica os carimbos de data/hora de modificação dos arquivos. Se um pré-requisito for mais recente que seu alvo, o make sabe que o alvo está desatualizado e executa a receita correspondente para recriá-lo. Essa abordagem é extremamente eficiente, pois garante que apenas as partes necessárias de uma pipeline complexa sejam reexecutadas após uma alteração, economizando tempo computacional significativo.[23]

Mais importante, um Makefile bem escrito serve como documentação executável para todo o projeto. Ele encapsula toda a lógica do fluxo de trabalho, desde o download dos dados brutos até a geração do manuscrito final. Um novo usuário (ou o pesquisador original, meses depois) pode clonar o repositório e, idealmente, executar um único comando — make — para reproduzir toda a análise do zero.[7] Isso atinge o objetivo final da proveniência executável e torna o projeto transparente, auditável e verdadeiramente reprodutível.

Seção 3: O Conjunto de Dados como um Cidadão de Primeira Classe: Documentação e Benchmarking

Em muitos projetos computacionais, o código-fonte recebe a maior parte da atenção em

termos de documentação e rigor, enquanto o conjunto de dados é tratado como um artefato secundário. Para uma pesquisa verdadeiramente reprodutível e reutilizável, o conjunto de dados deve ser elevado ao status de um resultado de pesquisa de primeira classe, com documentação formal e contextualização por meio de benchmarks rigorosos.

3.1 Documentação Formal com "Datasheets for Datasets"

A falta de documentação padronizada para conjuntos de dados é uma das principais causas de falhas na reprodutibilidade e no uso indevido de modelos de aprendizado de máquina.[27] Para resolver essa lacuna, Gebru et al. propuseram o framework "Datasheets for Datasets", inspirado nas folhas de dados abrangentes que acompanham os componentes eletrônicos.[29] Um datasheet para um conjunto de dados documenta sua motivação, composição, processo de coleta, usos recomendados e limitações.

Os componentes principais de um datasheet, juntamente com as perguntas que os criadores de conjuntos de dados devem abordar, incluem [27]:

- **Motivação:** Para que finalidade o conjunto de dados foi criado? Havia uma tarefa específica em mente? Quem financiou a criação?
- **Composição:** Que tipo de dados o conjunto de dados contém (por exemplo, imagens, texto, tabelas)? Quantas instâncias existem? O conjunto de dados contém informações que podem ser consideradas confidenciais?
- **Processo de Coleta:** Como os dados foram coletados? De qual fonte? Quem esteve envolvido no processo de coleta de dados e como foram compensados? O consentimento informado foi obtido dos sujeitos dos dados?
- **Pré-processamento/Limpeza/Rotulagem:** Os dados foram pré-processados ou limpos? Como? Os dados foram rotulados e, em caso afirmativo, por quem e por meio de qual processo?
- **Usos:** Para quais tarefas o conjunto de dados deve ser usado? Já foi usado em outros trabalhos? Existem tarefas para as quais o conjunto de dados não deve ser usado?
- **Distribuição:** O conjunto de dados será distribuído? Onde e como? Existem restrições de licenciamento ou uso?
- **Manutenção:** Quem é responsável por manter o conjunto de dados? Com que frequência ele será atualizado?

Crucialmente, a criação de um datasheet não deve ser vista como uma tarefa de documentação post-hoc. Os autores do framework enfatizam que o processo destina-se a "incentivar a reflexão cuidadosa".[31] Ao se envolver com questões sobre motivação, ética de coleta e potenciais vieses

durante o processo de criação do conjunto de dados, os pesquisadores podem identificar e mitigar problemas em um estágio inicial. Por exemplo, um pesquisador pode perceber que sua estratégia de coleta está introduzindo um viés de amostragem ou que precisa de um consentimento mais explícito para um determinado uso dos dados. Dessa forma, o datasheet transcende a mera documentação para se tornar uma ferramenta ativa no design da pesquisa e na validação ética do próprio conjunto de dados. A recomendação, portanto, é criar o datasheet em paralelo com a coleta e o processamento dos dados, e não como uma etapa final antes da publicação.

3.2 O Papel dos Benchmarks: Contextualizando o Desempenho

Para que um novo algoritmo ou método seja considerado uma contribuição significativa, seu desempenho deve ser rigorosamente avaliado e contextualizado em relação ao estado da arte. Isso requer benchmarking em uma coleção diversificada de conjuntos de dados, incluindo instâncias do mundo real e sintéticas.[32]

- **Conjuntos de Dados do Mundo Real:** Utilizar coleções de grafos bem estabelecidas, como as dos desafios de implementação DIMACS ou do benchmark LDBC Graphalytics, é crucial. Esses conjuntos de dados, provenientes de domínios como redes sociais, conhecimento e biologia, capturam a complexidade e a irregularidade encontradas em aplicações práticas.[32]
- **Conjuntos de Dados Sintéticos:** Os dados do mundo real devem ser complementados com dados gerados sinteticamente. Os geradores de grafos sintéticos permitem o controle sistemático sobre as propriedades do grafo (por exemplo, tamanho, densidade, estrutura de comunidade), possibilitando a avaliação do desempenho do algoritmo em diferentes escalas e sob condições variadas.

Uma análise da literatura revela uma evolução nos padrões de benchmarking. Os primeiros trabalhos frequentemente usavam modelos de grafos aleatórios simples. Modelos mais sofisticados, como o **LFR (Lancichinetti–Fortunato–Radicchi)**, foram desenvolvidos para incorporar propriedades realistas, como distribuições de lei de potência para graus de nós e tamanhos de comunidades.[37] No entanto, implementações posteriores do LFR foram consideradas lentas e, às vezes, produziam resultados inconsistentes, prejudicando a reprodutibilidade.[39]

Essa deficiência levou ao desenvolvimento de geradores de benchmark de próxima geração, como o **ABCD (Artificial Benchmark for Community Detection)**. O ABCD foi projetado explicitamente para superar as limitações do LFR, oferecendo maior velocidade, escalabilidade, tratabilidade teórica e um parâmetro de mistura mais intuitivo que evita a

geração de "anti-comunidades" indesejáveis.[39] Outro modelo fundamental é o

SBM (Stochastic Block Model) e sua variante, o **Planted Partition Model**, que são inestimáveis para a análise teórica e para testar a capacidade de um algoritmo de recuperar estruturas de comunidade bem definidas com probabilidades de conexão intra e intercomunidade controláveis.[40]

Essa progressão de modelos de benchmark simples para modelos mais realistas e, finalmente, para modelos realistas que também são eficientes e reprodutíveis, reflete um amadurecimento do campo. Não é mais suficiente que um benchmark *pareça* realista; ele deve ser utilizável, escalável e, acima de tudo, reprodutível. Ao selecionar benchmarks para um novo estudo, os pesquisadores devem priorizar geradores modernos, bem documentados e eficientes, como o ABCD, para garantir que seus resultados experimentais sejam robustos, facilmente verificáveis e construídos sobre uma base metodológica sólida.

Seção 4: Garantindo o Rigor em Experimentos Computacionais

A apresentação dos resultados de experimentos computacionais é tão crítica quanto os próprios resultados. Uma análise de desempenho superficial ou estatisticamente falha pode levar a conclusões errôneas e prejudicar a credibilidade da pesquisa. Esta seção detalha uma metodologia robusta para a análise visual e estatística do desempenho de algoritmos, garantindo que as comparações sejam justas, defensáveis e informativas.

4.1 As Armadilhas do Benchmarking Ingênuo

A prática de relatar apenas uma única métrica de resumo, como o desempenho médio ou o melhor resultado de várias execuções, é inerentemente falha para algoritmos estocásticos. Tais abordagens podem ser enganosas, pois ocultam a variabilidade do desempenho e são suscetíveis a vieses.[44] Além disso, a comunidade de pesquisa em seleção de algoritmos identificou falhas metodológicas mais sutis que podem levar a conclusões excessivamente otimistas. Isso inclui a falha da avaliação "leave-instance-out", onde a semelhança entre instâncias do mesmo problema em conjuntos de treinamento e teste pode levar a uma superestimação do desempenho, e o uso de métricas sensíveis à escala (como o erro absoluto), que podem ser confundidas com a dificuldade inerente do problema.[46] Uma abordagem rigorosa deve ir além das médias e empregar ferramentas que capturem a

distribuição completa do desempenho.

4.2 Visualizando o Desempenho: Perfis de Desempenho e Gráficos de Tempo para o Alvo

A visualização é uma ferramenta poderosa para comparar o comportamento de algoritmos. Duas técnicas padrão-ouro para benchmarking de otimização são os Perfis de Desempenho e os Gráficos de Tempo para o Alvo (TTT).

Perfis de Desempenho (Dolan & Moré)
Os perfis de desempenho oferecem uma maneira de comparar vários solucionadores em um grande conjunto de problemas de teste.51 A metodologia é a seguinte:

1. **Definir uma Métrica de Desempenho:** Para cada problema p em um conjunto de problemas P e cada solucionador s em um conjunto de solucionadores S, meça uma métrica de desempenho, como o tempo de CPU ou o número de avaliações de função, denotado por $t_{p,s}$.
2. **Calcular a Razão de Desempenho:** Para cada problema p, identifique o melhor desempenho de qualquer solucionador: $\min_s\{t_{p,s}\}$. A razão de desempenho para cada solucionador é então calculada como $r_{p,s} = \min_s\{t_{p,s}\} t_{p,s}$. Se um solucionador s não conseguir resolver o problema p, $r_{p,s}$ é definido como ∞.
3. Construir a Função de Distribuição Cumulativa: O perfil de desempenho para um solucionador s é a função de distribuição cumulativa (CDF) da razão de desempenho, $\rho_s(\tau)$, que é a fração de problemas para os quais a razão de desempenho do solucionador s está dentro de um fator τ do melhor:

   $$\rho_s(\tau) = \frac{1}{|P|} |\{p \in P | r_{p,s} \leq \tau\}|$$

A interpretação de um perfil de desempenho fornece insights em duas dimensões chave:

- **Eficiência:** O valor de $\rho_s(1)$ representa a fração de problemas para os quais o solucionador s foi o melhor (ou empatou para o melhor). Um valor mais alto no eixo y em $\tau=1$ indica um solucionador mais eficiente.
- **Robustez:** O limite do perfil à medida que $\tau \to \infty$ (na prática, o valor no extremo direito do gráfico) representa a fração de problemas que o solucionador s conseguiu resolver. Um valor mais alto indica um solucionador mais robusto.[56]

Gráficos de Tempo para o Alvo (TTT)
Os gráficos TTT, ou distribuições de tempo de execução, são usados para caracterizar o desempenho de um único algoritmo estocástico em uma única instância de problema.60 Eles são construídos executando o algoritmo

N vezes com sementes aleatórias diferentes e registrando o tempo necessário para atingir um valor-alvo de solução predefinido. A Função de Distribuição Cumulativa Empírica (ECDF) desses tempos é então plotada, mostrando a probabilidade de o algoritmo atingir o alvo dentro de um determinado tempo de execução.

Uma característica crucial de ambas as técnicas de visualização é a sua capacidade de lidar com **dados censurados**, ou seja, execuções malsucedidas em que um algoritmo não consegue resolver um problema ou atingir um alvo dentro de um orçamento de tempo predefinido.[63] Essas falhas não devem ser descartadas; elas são dados essenciais que quantificam a robustez do algoritmo. No framework ECDF, os dados censurados são tratados naturalmente. A ECDF,

F(t), representa a proporção de execuções que *tiveram sucesso* até o tempo t. Se k de um total de N execuções forem bem-sucedidas, a curva ECDF subirá a cada sucesso, mas acabará por se estabilizar (saturar) a uma altura final de k/N. Essa altura de saturação não é uma falha do gráfico; é uma medida quantitativa direta da **taxa de sucesso** ou robustez do algoritmo. Um algoritmo que resolve o problema em todas as execuções terá uma curva que atinge 1.0, enquanto um que tem sucesso em 80% das vezes terá uma curva que satura em 0.8.[64] Assim, esses gráficos combinam elegantemente as dimensões de eficiência (a rapidez com que a curva sobe) e robustez (a altura que a curva atinge).

4.3 Validação Estatística: Uma Hierarquia de Evidências

Enquanto as visualizações fornecem uma visão geral intuitiva, a validação estatística é necessária para fazer afirmações rigorosas sobre as diferenças de desempenho. Conforme defendido por Demšar (2006), os testes não paramétricos são geralmente preferíveis para comparar algoritmos, pois as suposições subjacentes aos testes paramétricos (como a normalidade das distribuições de desempenho) raramente são satisfeitas na prática.[65]

Um fluxo de trabalho estatístico robusto para comparar múltiplos algoritmos em múltiplos conjuntos de dados segue uma hierarquia de evidências:

1. **Teste Omnibus para Diferença Global (Teste de Friedman):** O primeiro passo é determinar se existe *alguma* diferença estatisticamente significativa entre o grupo de algoritmos. O teste de Friedman é o análogo não paramétrico da ANOVA de medidas repetidas. Ele funciona classificando o desempenho dos algoritmos em cada conjunto de dados e, em seguida, comparando os postos médios entre os algoritmos. Uma rejeição da hipótese nula (de que todos os algoritmos têm desempenho equivalente) indica que existem diferenças significativas que merecem uma investigação mais aprofundada.[66]
2. **Testes Post-Hoc para Comparações Par a Par (Teste de Nemenyi):** Se o teste de

Friedman for significativo, testes post-hoc são usados para identificar *quais pares específicos* de algoritmos diferem significativamente. O teste de Nemenyi é um teste post-hoc comum que compara os postos médios de todos os pares de algoritmos. Ele calcula uma "Diferença Crítica" (CD); se a diferença nos postos médios para qualquer par de algoritmos exceder o CD, a diferença é considerada estatisticamente significativa.[67] Os resultados podem ser visualizados de forma eficaz usando diagramas de diferença crítica.

3. **Comparação Direta de Dois Algoritmos (Teste de postos sinalizados de Wilcoxon):** Ao comparar apenas dois algoritmos, um teste mais poderoso e direto é o teste de postos sinalizados de Wilcoxon. Este é o análogo não paramétrico do teste t pareado e avalia se as diferenças de desempenho entre os dois algoritmos em vários conjuntos de dados são consistentemente positivas ou negativas.[65]

Quantificando a Magnitude da Diferença: Tamanho do Efeito
A significância estatística (ou seja, um valor-p baixo) não informa sobre a magnitude da diferença de desempenho. Para isso, uma medida de tamanho do efeito é necessária. A estatística Vargha-Delaney A12 é uma medida de tamanho do efeito não paramétrica e intuitiva. Ela quantifica a probabilidade de que um algoritmo A produza um resultado melhor do que um algoritmo B em uma instância de problema selecionada aleatoriamente. Um valor A12 de 0.5 indica um desempenho equivalente, enquanto um valor de 0.8 significa que o algoritmo A superará o algoritmo B em 80% das vezes. Isso fornece uma medida prática e facilmente interpretável da superioridade de um algoritmo sobre outro, complementando os resultados dos testes de hipóteses.75

Etapa	Teste / Métrica	Hipótese Nula / Propósito	Interpretação
1. Comparação Global	Teste de Friedman	Não há diferença no desempenho médio entre qualquer um dos algoritmos.	Se o valor-p < α, rejeite a hipótese nula. Prossiga para as comparações par a par para identificar onde estão as diferenças.
2. Comparação Par a Par	Teste Post-Hoc de Nemenyi	Não há diferença de desempenho entre um par específico de algoritmos.	Se a diferença nos postos médios > Diferença Crítica (CD), a diferença é estatisticamente significativa.

3. Quantificar a Diferença	Estatística A12 de Vargha-Delaney	Quantificar a magnitude da diferença entre dois algoritmos.	Medida de tamanho do efeito probabilística. A12 = 0.7 indica que o Algoritmo 1 tem 70% de probabilidade de superar o Algoritmo 2.

Seção 5: Caminhos para a Publicação: Disseminando Código e Dados

Na pesquisa científica moderna, a publicação não se limita mais ao manuscrito. O código-fonte e os conjuntos de dados que sustentam a pesquisa são, eles próprios, contribuições acadêmicas valiosas. A publicação formal desses artefatos em locais especializados e citáveis é crucial para garantir o crédito, a descoberta e a longevidade do trabalho de pesquisa.

5.1 Publicando Software de Pesquisa: O Journal of Open Source Software (JOSS)

O Journal of Open Source Software (JOSS) é um periódico acadêmico revisado por pares, projetado especificamente para oferecer crédito acadêmico formal para software de pesquisa.[78] Diferentemente dos periódicos tradicionais, o foco da revisão do JOSS não está em novos resultados de pesquisa, mas na qualidade do próprio software.[81]

Os critérios de submissão para o JOSS são rigorosos e refletem as melhores práticas no desenvolvimento de software de código aberto [81]:

- **Esforço Acadêmico Substancial:** O software deve representar um esforço significativo, geralmente equivalente a pelo menos três meses de trabalho de um indivíduo. Submissões com menos de 1000 linhas de código são examinadas de perto.[81]
- **Licença de Código Aberto:** O software deve ser distribuído sob uma licença de código

aberto aprovada pela OSI (Open Source Initiative).
- **Repositório Público:** O código deve ser hospedado em um repositório público (por exemplo, GitHub) que permita a navegação do código, o rastreamento de problemas e contribuições da comunidade.
- **Documentação de Alta Qualidade:** A documentação abrangente é um requisito fundamental, explicando como instalar, usar e contribuir para o software.
- **Testes Automatizados:** A presença de uma suíte de testes (por exemplo, usando integração contínua) para verificar a correção do software é um critério de revisão chave.[81]
- **Formato do paper.md:** A submissão consiste em um arquivo paper.md curto e em formato Markdown. Este arquivo deve incluir um título, autores e afiliações, um resumo descrevendo o propósito e a funcionalidade do software, uma "Declaração de Necessidade" que contextualiza o software no campo de pesquisa e referências importantes.[81]

5.2 Publicando Conjuntos de Dados de Pesquisa: Data in Brief

De forma análoga ao JOSS para software, periódicos como *Data in Brief* oferecem um local para a publicação revisada por pares de artigos de dados.[83] O objetivo desses artigos é descrever conjuntos de dados de forma detalhada, tornando-os mais fáceis de encontrar, citar e reutilizar. O foco é estritamente descritivo; os artigos não devem conter análises, interpretações ou conclusões sobre os dados.[83]

A estrutura de um artigo no *Data in Brief* segue um modelo rigoroso [84]:

- **Resumo e Tabela de Especificações:** Um resumo puramente descritivo e uma tabela que detalha as especificações técnicas do conjunto de dados (por exemplo, tipo de dados, formato, como foram adquiridos).
- **Seção "Valor dos Dados" (Value of the Data):** Esta é uma seção crucial que exige que os autores articulem, em 3 a 5 pontos, por que o conjunto de dados é valioso para a comunidade científica em geral. Isso incentiva os autores a pensar além de seu próprio caso de uso e a destacar o potencial de reutilização dos dados.[83]
- **Acessibilidade dos Dados:** Um requisito não negociável é que os dados devem ser depositados em um repositório público e confiável (por exemplo, Mendeley Data, Figshare, Zenodo). O artigo deve fornecer um link direto e um identificador persistente (como um DOI) para o conjunto de dados, garantindo o acesso permanente.[83]

A ascensão de periódicos como JOSS e *Data in Brief* sinaliza uma mudança cultural fundamental na publicação acadêmica. Tradicionalmente, código e dados eram relegados ao status de "material suplementar", seu valor intrinsecamente ligado à narrativa de um único

artigo de pesquisa. Este modelo não apenas desvaloriza o imenso esforço necessário para criar software e conjuntos de dados de alta qualidade, mas também limita sua descoberta e reutilização. Ao cunhar DOIs separados para os artigos de software e dados, esses periódicos elevam os artefatos de pesquisa ao status de produtos de pesquisa de primeira classe, independentes e citáveis.[78] Isso cria um ciclo virtuoso: os pesquisadores são incentivados a produzir artefatos bem documentados e de alta qualidade porque agora recebem crédito acadêmico direto por eles, o que, por sua vez, acelera o progresso científico ao tornar esses blocos de construção fundamentais mais acessíveis e confiáveis para toda a comunidade.

Seção 6: Recomendações Integradas e Lista de Verificação do Projeto

Esta seção final sintetiza as discussões e recomendações das seções anteriores em uma lista de verificação acionável. Esta lista de verificação foi projetada para orientar os pesquisadores na preparação de seus artefatos de projeto — código, conjunto de dados, experimentos e artigo — para garantir que eles atendam aos mais altos padrões de reprodutibilidade e estejam prontos para publicação e reutilização.

Lista de Verificação de Reprodutibilidade do Projeto

Artefato: Código-Fonte

- **[ ] Controle de Versão:** Todo o código-fonte, scripts, arquivos de configuração e documentação baseada em texto estão sendo rastreados em um repositório Git?
  - *Referência: Seção 2.1*
- **[ ] Estrutura do Repositório:** O repositório está organizado de forma lógica, com diretórios separados para código-fonte (src), dados (data), saídas (results), etc.?
  - *Referência: Princípios gerais de organização de projetos.*
- **[ ] Gerenciamento de Dependências:** O ambiente computacional está definido explicitamente?
  - **[ ] Estratégia Híbrida:** O environment.yml do Conda é usado para o interpretador Python e dependências não-Python?
  - **[ ] Bloqueio Determinístico:** O pyproject.toml e o poetry.lock do Poetry são usados para todas as dependências Python?
  - *Referência: Seção 2.2*
- **[ ] Documentação do Código:** O código está bem comentado? Existe um arquivo

README.md de alto nível que explica o propósito do projeto, como instalá-lo e como executar a análise?
    - *Referência: Pilar 5 (Documentação)*
- **[ ] Licenciamento:** O repositório inclui um arquivo LICENSE com uma licença de código aberto aprovada pela OSI?
    - *Referência: Seção 5.1*
- **[ ] Testes:** Existe uma suíte de testes automatizados para verificar a correção do código? A integração contínua está configurada para executar esses testes automaticamente?
    - *Referência: Seção 5.1*

Artefato: Conjunto de Dados

- **[ ] Controle de Versão de Dados:** Os arquivos de dados grandes estão sendo rastreados com uma ferramenta apropriada?
    - **[ ] DVC:** O DVC é usado para rastrear dados e modelos, com as pipelines definidas em dvc.yaml?
    - **[ ] Git LFS:** (Para projetos mais simples) O Git LFS está configurado para os tipos de arquivo apropriados?
    - *Referência: Seção 2.1*
- **[ ] Documentação do Conjunto de Dados:** Um "Datasheet for Datasets" foi criado para documentar a motivação, composição, coleta e usos pretendidos do conjunto de dados?
    - *Referência: Seção 3.1*
- **[ ] Acessibilidade e Persistência:** O conjunto de dados final foi depositado em um repositório de dados público e confiável (por exemplo, Zenodo, Figshare) e recebeu um identificador persistente (DOI)?
    - *Referência: Seção 5.2*
- **[ ] Validação de Dados:** Os formatos de dados de entrada e saída são validados usando um esquema, como JSON Schema, para garantir a integridade e a conformidade estrutural?
    - Referência: [88]

Artefato: Fluxo de Trabalho Experimental

- **[ ] Automação de Ponta a Ponta:** Um Makefile (ou ferramenta de automação de fluxo de trabalho similar) foi criado para automatizar toda a pipeline de análise, desde os dados brutos até as figuras e tabelas finais?
    - *Referência: Seção 2.3*
- **[ ] Reprodutibilidade com um Único Comando:** É possível regenerar todos os resultados executando um único comando (por exemplo, make all)?
    - *Referência: Seção 1.1 (Proveniência Executável)*
- **[ ] Sementes Aleatórias:** Para todas as análises que envolvem estocasticidade, a semente do gerador de números aleatórios está fixada e registrada para garantir resultados idênticos em execuções repetidas?
    - *Referência: Regra 6 (Dez Regras Simples)*

- **[ ] Seleção de Benchmark:** Os algoritmos foram avaliados em uma combinação de conjuntos de dados do mundo real bem estabelecidos e geradores de grafos sintéticos modernos e reprodutíveis (por exemplo, ABCD)?
    - *Referência: Seção 3.2*

Artefato: Análise e Publicação

- **[ ] Análise Visual Robusta:** Os resultados de desempenho são visualizados usando Perfis de Desempenho (para múltiplos solucionadores) e/ou Gráficos de Tempo para o Alvo (para algoritmos estocásticos), representando corretamente as execuções malsucedidas (dados censurados)?
    - *Referência: Seção 4.2*
- **[ ] Validação Estatística Rigorosa:** O fluxo de trabalho estatístico recomendado foi seguido?
    - **[ ] Teste de Friedman:** Usado para a comparação global de múltiplos algoritmos.
    - **[ ] Teste de Nemenyi:** Usado para comparações par a par post-hoc.
    - **[ ] Teste de Wilcoxon:** Usado para a comparação direta de dois algoritmos.
    - **[ ] Tamanho do Efeito A12:** Calculado para quantificar a magnitude das diferenças de desempenho.
    - *Referência: Seção 4.3*
- **[ ] Publicação de Artefatos:** Foram preparadas submissões para periódicos apropriados para os artefatos?
    - **[ ] JOSS:** Um paper.md foi preparado para o software de pesquisa.
    - **[ ] Data in Brief:** Um artigo de dados foi preparado para o conjunto de dados.
    - *Referência: Seção 5*

Ao abordar sistematicamente cada item desta lista de verificação, os pesquisadores podem aumentar significativamente a transparência, a robustez e o impacto de seu trabalho computacional, garantindo que suas contribuições não sejam apenas publicadas, mas também compreendidas, validadas e construídas pela comunidade científica.

Referências citadas

1. The five pillars of computational reproducibility: Bioinformatics and beyond - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/publication/371671830_The_five_pillars_of_computational_reproducibility_Bioinformatics_and_beyond
2. The five pillars of computational reproducibility: bioinformatics and beyond - PubMed, acessado em agosto 29, 2025, https://pubmed.ncbi.nlm.nih.gov/37870287/
3. five pillars of computational reproducibility: bioinformatics and ..., acessado em agosto 29, 2025, https://academic.oup.com/bib/article/24/6/bbad375/7326135
4. The five pillars of computational reproducibility: bioinformatics and beyond - OUCI, acessado em agosto 29, 2025, https://ouci.dntb.gov.ua/en/works/7BKpE5Z4/

5. The five pillars of computational reproducibility. - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/figure/The-five-pillars-of-computational-reproducibility_fig1_374922562
6. academic.oup.com, acessado em agosto 29, 2025, https://academic.oup.com/bib/article/24/6/bbad375/7326135#:~:text=These%20include%20(1)%20literate%20programming,easily%2C%20long%20into%20the%20future.
7. Ten Simple Rules for Reproducible Computational Research - PMC, acessado em agosto 29, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC3812051/
8. Git LFS and DVC: The Ultimate Guide to Managing Large Artifacts in ..., acessado em agosto 29, 2025, https://medium.com/@pablojusue/git-lfs-and-dvc-the-ultimate-guide-to-managing-large-artifacts-in-mlops-c1c926e6c5f4
9. Support for Git Repository Packages which have LFS · Issue #8723 · python-poetry/poetry, acessado em agosto 29, 2025, https://github.com/python-poetry/poetry/issues/8723
10. Why Git LFS Is Not Good Practice for AI Model Weights and Why You Should Use DVC Instead (Demo with Azure Data Lake Storage 2) | by Neel Shah | Medium, acessado em agosto 29, 2025, https://medium.com/@neeldevenshah/why-git-lfs-is-not-good-practice-for-ai-model-weights-and-why-you-should-use-dvc-instead-demo-with-3903a7ae68f5
11. What is the best way to do data version control for Machine Learning models - MATLAB Answers - MathWorks, acessado em agosto 29, 2025, https://www.mathworks.com/matlabcentral/answers/2067961-what-is-the-best-way-to-do-data-version-control-for-machine-learning-models
12. Difference between git-lfs and dvc - Stack Overflow, acessado em agosto 29, 2025, https://stackoverflow.com/questions/58541260/difference-between-git-lfs-and-dvc
13. DVC vs Git vs Git LFS: ML Reproducibility - Censius, acessado em agosto 29, 2025, https://censius.ai/blogs/dvc-vs-git-and-git-lfs-in-machine-learning-reproducibility
14. Get Started with DVC | Data Version Control, acessado em agosto 29, 2025, https://dvc.org/doc/start
15. Building Reproducible ML Pipelines with DVC and GitLab | by Ramkrushna Maheshwar, acessado em agosto 29, 2025, https://medium.com/@maheshwar.ramkrushna/building-reproducible-ml-pipelines-with-dvc-and-gitlab-0ab768eadb89
16. Data Version Control With Python and DVC, acessado em agosto 29, 2025, https://realpython.com/python-data-version-control/
17. iterative/dvc: Data Versioning and ML Experiments - GitHub, acessado em agosto 29, 2025, https://github.com/iterative/dvc
18. Conda vs Poetry in Python - GeeksforGeeks, acessado em agosto 29, 2025,

https://www.geeksforgeeks.org/python/conda-vs-poetry-in-python/
19. Poetry vs Conda : r/Python - Reddit, acessado em agosto 29, 2025, https://www.reddit.com/r/Python/comments/t4g7xf/poetry_vs_conda/
20. python - Does it make sense to use Conda + Poetry? - Stack Overflow, acessado em agosto 29, 2025, https://stackoverflow.com/questions/70851048/does-it-make-sense-to-use-conda-poetry
21. Managing Python Dependencies with Poetry vs Conda & Pip - Exxact Corporation, acessado em agosto 29, 2025, https://www.exxactcorp.com/blog/Deep-Learning/managing-python-dependencies-with-poetry-vs-conda-pip
22. Conda and Poetry: A Harmonious Fusion | by Henrique Malta - Stackademic, acessado em agosto 29, 2025, https://blog.stackademic.com/conda-and-poetry-a-harmonious-fusion-8116895b6380
23. Reproducibility with Make - The Turing Way, acessado em agosto 29, 2025, https://book.the-turing-way.org/reproducible-research/make
24. Use Makefiles to Manage, Automate, and Reproduce Projects - Tilburg Science Hub, acessado em agosto 29, 2025, https://tilburgsciencehub.com/topics/automation/automation-tools/make-and-makefiles/what-are-makefiles/
25. Reproducible bioinformatics pipelines using Make, acessado em agosto 29, 2025, http://byronjsmith.com/make-bml/
26. Automation with make - Riffomonas, acessado em agosto 29, 2025, https://riffomonas.org/reproducible_research/make/
27. Datasheets for Datasets - Morgan Klaus Scheuerman, acessado em agosto 29, 2025, https://www.morgan-klaus.com/readings/datasheets-for-datasets.html
28. Datasheets for Datasets - Microsoft, acessado em agosto 29, 2025, https://www.microsoft.com/en-us/research/wp-content/uploads/2019/01/1803.09010.pdf
29. Datasheets for Datasets - AI Now Institute, acessado em agosto 29, 2025, https://ainowinstitute.org/publications/datasheets-for-datasets
30. [1803.09010] Datasheets for Datasets - arXiv, acessado em agosto 29, 2025, https://arxiv.org/abs/1803.09010
31. Datasheets for Datasets - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/1803.09010
32. Graph Partitioning and Graph Clustering - David A. Bader, acessado em agosto 29, 2025, https://davidbader.net/publication/2013-bmsw/2013-bmsw.pdf
33. LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis ..., acessado em agosto 29, 2025, https://www.vldb.org/pvldb/vol9/p1317-iosup.pdf
34. Partitioning Trillion-edge Graphs in Minutes - OSTI.GOV, acessado em agosto 29, 2025, https://www.osti.gov/servlets/purl/1484927
35. LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms - TU Delft Research Portal, acessado em agosto 29, 2025, https://research.tudelft.nl/files/66845401/p1317_iosup.pdf

36. Scalable Graph Partitioning for Distributed Graph Processing - Universität Stuttgart, acessado em agosto 29, 2025, https://www2.informatik.uni-stuttgart.de/bibliothek/ftp/ncstrl.ustuttgart_fi/DIS-2019-03/DIS-2019-03.pdf
37. Lancichinetti–Fortunato–Radicchi benchmark - Wikipedia, acessado em agosto 29, 2025, https://en.wikipedia.org/wiki/Lancichinetti%E2%80%93Fortunato%E2%80%93Radicchi_benchmark
38. LFR_benchmark_graph — NetworkX 3.5 documentation, acessado em agosto 29, 2025, https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.LFR_benchmark_graph.html
39. Artificial Benchmark for Community Detection (ABCD)—Fast random ..., acessado em agosto 29, 2025, https://www.cambridge.org/core/services/aop-cambridge-core/content/view/453FE29A1FA3C1798B0EC116587FE422/S2050124220000454a.pdf/artificial-benchmark-for-community-detection-abcdfast-random-graph-model-with-community-structure.pdf
40. Stochastic block model - Wikipedia, acessado em agosto 29, 2025, https://en.wikipedia.org/wiki/Stochastic_block_model
41. Improved Community Detection using Stochastic Block Models - arXiv, acessado em agosto 29, 2025, https://arxiv.org/html/2408.10464v2
42. Structural Entropy of the Stochastic Block Models - MDPI, acessado em agosto 29, 2025, https://www.mdpi.com/1099-4300/24/1/81
43. SBGD: Improving Graph Diffusion Generative Model via Stochastic Block Diffusion - arXiv, acessado em agosto 29, 2025, https://arxiv.org/html/2508.14352v1
44. On the Estimation of the Expected Performance of a Metaheuristic ..., acessado em agosto 29, 2025, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=800a5f55ce2bb62892c00834779d4e83fd2fa759
45. How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/publication/226547420_How_to_assess_and_report_the_performance_of_a_stochastic_algorithm_on_a_benchmark_problem_Mean_or_best_result_on_a_number_of_runs
46. The Pitfalls of Benchmarking in Algorithm Selection: What We Are Getting Wrong - arXiv, acessado em agosto 29, 2025, https://arxiv.org/html/2505.07750v1
47. The Pitfalls of Benchmarking in Algorithm Selection: What We ... - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/2505.07750
48. The Pitfalls of Benchmarking in Algorithm Selection: What We Are Getting Wrong, acessado em agosto 29, 2025, https://www.researchgate.net/publication/393651527_The_Pitfalls_of_Benchmarking_in_Algorithm_Selection_What_We_Are_Getting_Wrong

49. Rethinking Metaheuristics: Unveiling the Myth of “Novelty” in Metaheuristic Algorithms, acessado em agosto 29, 2025, https://www.mdpi.com/2227-7390/13/13/2158
50. Metaheuristics exposed: Unmasking the design pitfalls of arithmetic optimization algorithm in benchmarking | Request PDF - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/publication/380380954_Metaheuristics_exposed_Unmasking_the_design_pitfalls_of_arithmetic_optimization_algorithm_in_benchmarking
51. Benchmarking optimization software with performance profiles. (Journal Article) | OSTI.GOV, acessado em agosto 29, 2025, https://www.osti.gov/biblio/943229
52. Benchmarking Optimization Software with Performance Profiles, acessado em agosto 29, 2025, https://www.researchgate.net/publication/1853672_Benchmarking_Optimization_Software_with_Performance_Profiles
53. Performance Profile Benchmarking Tool - Tangi Migot, acessado em agosto 29, 2025, https://tmigot.github.io/posts/2024/06/teaching/
54. Dolan, E.D. and Moré, J.J. (2002) Benchmarking Optimization Software with Performance Profiles. Mathematical Programming, 91, 201-213. - References, acessado em agosto 29, 2025, https://www.scirp.org/reference/referencespapers?referenceid=2663487
55. benchmarking derivative-free optimization algorithms - CiteSeerX, acessado em agosto 29, 2025, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=90319b0b2a3c700015303fa2f30dcdc22f081ac1
56. Introduction to Performance Profile | Abel Soares Siqueira, acessado em agosto 29, 2025, https://abelsiqueira.com/blog/2017-05-09-introduction-to-performance-profile/
57. A note on performance profiles for benchmarking software - ePubs, acessado em agosto 29, 2025, https://epubs.stfc.ac.uk/manifestation/20477017/RAL-P-2015-004.pdf
58. A Note on Performance Profiles for Benchmarking Software | Request PDF - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/publication/310824952_A_Note_on_Performance_Profiles_for_Benchmarking_Software
59. JuliaSmoothOptimizers/BenchmarkProfiles.jl: Performance and data profiles - GitHub, acessado em agosto 29, 2025, https://github.com/JuliaSmoothOptimizers/BenchmarkProfiles.jl
60. Extending time-to-target plots to multiple instances and targets ... - UFF, acessado em agosto 29, 2025, http://www.dcc.ic.uff.br/~celso/artigos/mtttplot-mic2017.pdf
61. tttplots-compare: a perl program to compare time-to-target plots or general runtime distributions of randomized algorithms | Request PDF - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/publication/271741508_tttplots-compare_a_perl_pr

ogram_to_compare_time-to-target_plots_or_general_runtime_distributions_of_randomized_algorithms
62. tttplots: a perl program to create time-to-target plots - UFF, acessado em agosto 29, 2025, http://profs.ic.uff.br/~celso/artigos/tttplots.pdf
63. Best Practices for Comparing Optimization Algorithms - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/1709.08242
64. Volume 13, Issue 4, acessado em agosto 29, 2025, https://evolution.sigevo.org/issues/HTML/sigevolution-13-4/index.htm
65. Statistical Comparisons of Classifiers over Multiple Data Sets, acessado em agosto 29, 2025, https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf
66. A Graphical Approach for Friedman Test: Moments Approach - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/2202.09131
67. How to Show that Your Model is Better: A Basic Guide to Statistical Hypothesis Testing, acessado em agosto 29, 2025, https://lab.rivas.ai/?p=2665
68. Friedman test and Nemenyi post-hoc test when comparing the time spent... - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/figure/Friedman-test-and-Nemenyi-post-hoc-test-when-comparing-the-time-spent-by-the-different-AO_fig5_346555784
69. Nemenyi test - Wikipedia, acessado em agosto 29, 2025, https://en.wikipedia.org/wiki/Nemenyi_test
70. Statistical Comparisons of Classifiers over Multiple Data Sets - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/publication/220320196_Statistical_Comparisons_of_Classifiers_over_Multiple_Data_Sets
71. Understanding the Wilcoxon Sign Test - Statistics Solutions, acessado em agosto 29, 2025, https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/wilcoxon-sign-test/
72. Wilcoxon signed-rank test - Wikipedia, acessado em agosto 29, 2025, https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test
73. Wilcoxon Signed Rank Test - GeeksforGeeks, acessado em agosto 29, 2025, https://www.geeksforgeeks.org/machine-learning/wilcoxon-signed-rank-test/
74. Statistics: 2.2 The Wilcoxon signed rank sum test - Statstutor, acessado em agosto 29, 2025, https://www.statstutor.ac.uk/resources/uploaded/wilcoxonsignedranktest.pdf
75. Common Language Effect Size - PS, acessado em agosto 29, 2025, https://peterstatistics.com/Terms/EffectSizes/CommonLanguageEffectSize.html
76. Wilcoxon Rank Sum Test and Vargha-Delaney Â12 Measure for the Traditional Metrics-based Models. - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/figure/Wilcoxon-Rank-Sum-Test-and-Vargha-Delaney-A12-Measure-for-the-Traditional-Metrics-based_tbl3_328210662
77. Average Vargha-Delaney effect sizes b A12 of each algorithm (in a row)... - ResearchGate, acessado em agosto 29, 2025, https://www.researchgate.net/figure/Average-Vargha-Delaney-effect-sizes-b-A12-of-each-algorithm-in-a-row-with-another_tbl3_349080081

78. About the Journal of Open Source Software, acessado em agosto 29, 2025, https://joss.theoj.org/about
79. Journal of Open Source Software — JOSS documentation, acessado em agosto 29, 2025, https://joss.readthedocs.io/
80. Journal of Open Source Software, acessado em agosto 29, 2025, https://joss.theoj.org/
81. Submitting a paper to JOSS — JOSS documentation, acessado em agosto 29, 2025, https://joss.readthedocs.io/en/latest/submitting.html
82. Journal of Open Source Software (JOSS): design and first-year review - PMC, acessado em agosto 29, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC7340488/
83. How to write a good Data in Brief article - Elsevier Researcher Academy, acessado em agosto 29, 2025, https://researcheracademy.elsevier.com/uploads/2024-07/Quick_guide_Data_in_brief.pdf
84. Datainbrief Template | PDF | Academic Publishing | Data - Scribd, acessado em agosto 29, 2025, https://www.scribd.com/document/375142518/Datainbrief-Template
85. Data in Brief and MethodsX: How to submit your Co-Submission - YouTube, acessado em agosto 29, 2025, https://www.youtube.com/watch?v=Ab9E0dnvVIs
86. Data in Brief template - Elsevier, acessado em agosto 29, 2025, https://legacyfileshare.elsevier.com/promis_misc/data-in-brief-article-template.docx
87. Data in Brief: How do I submit a data article? - YouTube, acessado em agosto 29, 2025, https://www.youtube.com/watch?v=La1eZKDx1f4
88. Creating your first schema - JSON Schema, acessado em agosto 29, 2025, https://json-schema.org/learn/getting-started-step-by-step
89. Optimizing JSON Schema for Scalable Computer Vision Pipelines | Medium, acessado em agosto 29, 2025, https://medium.com/@noel.benji/designing-scalable-json-schemas-for-computer-vision-pipelines-dcddf4e7a9f4
90. Need to Verify Your JSON Schema? Here's a Few Ways to Do It ..., acessado em agosto 29, 2025, https://zuplo.com/blog/verify-json-schema
91. Introducing JSON Schemas for AI Data Integrity - DEV Community, acessado em agosto 29, 2025, https://dev.to/stephenc222/introducing-json-schemas-for-ai-data-integrity-611