The Journal of Open Source Software

CSPBench: A Comprehensive Framework for Closest String Problem Benchmarking

Diego Grosmann [1]

1 Independent Researcher

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

The **Closest String Problem (CSP)** is a classical NP-hard optimization problem with many applications in computational biology, such as motif discovery, consensus sequence identification, and error correction in biological sequences. Formally, given a set of strings $S = \{s_1, \dots, s_n\}$ of length d over an alphabet Σ, CSP asks for a center string $x \in \Sigma^d$ that minimizes

$$\max_i \mathrm{HammingDistance}(x, s_i)$$

Recent literature provides context for heuristics and complexity results — for example, heuristic advances on gene sequences (Abdi et al., 2024) and fine-grained complexity limits and algorithms in specific regimes (Abboud et al., 2023). Despite this, a practical, extensible, and reproducible benchmarking framework that unifies algorithms, datasets, and interfaces has remained scarce. **CSPBench** fills this gap with a modular toolkit focused on side-by-side evaluation.

Statement of Need

Researchers face three recurring gaps: (i) scarce, apples-to-apples comparisons between heuristics and exact/baseline methods; (ii) a disconnect between theory and practice—complexity insights rarely arrive with implementations suitable for bioinformatics data; and (iii) limited reproducibility due to heterogeneous datasets, parameterizations, and missing resource tracking. Tooling to load FASTA data, generate synthetic instances, monitor runs, and archive results consistently remains essential.

CSPBench addresses these needs by combining: (a) a modular hexagonal architecture with an auto-registered algorithm plugin system; (b) baseline, heuristic, and exact implementations executed under identical conditions; (c) dataset support for FASTA, synthetic generation, and optional NCBI Entrez; (d) standardized metrics (max/avg/total Hamming, runtime) with structured metadata; and (e) CLI (menu e batch) e Web (beta) para fluxos distintos.

Related Work

Two recent reference points illustrate the landscape: a three-stage heuristic tailored to gene sequences (Abdi et al., 2024) and complexity limits with improved algorithms in specific regimes (Abboud et al., 2023). Both underscore the value of a practical benchmarking harness that unifies implementations, datasets, and consistent evaluation—precisely the niche addressed by CSPBench.

Software Overview

- Algorithms (auto-discovered via `@register_algorithm`): Baseline (greedy consensus), CSC (clustering + recombinação), BLF-GA (metaheurística híbrida), H2-CSP (heurística híbrida), DP-CSP (exato para instâncias pequenas).

- Datasets: geração sintética; leitura de FASTA em `DATASET_DIRECTORY`; download opcional via NCBI Entrez quando configurado.

- Execução: CLI (Typer) com menu interativo e modo batch; Web (FastAPI, beta) para datasets, algoritmos, batches e monitoramento.

- Métricas: distância de Hamming (máx/média/total), tempo de execução, metadados (tamanho, alfabeto, seed, paralelismo interno quando aplicável).

- Arquitetura: hexagonal (domain puro; application services; infrastructure; presentation; plugins em `algorithms/`).

Experiments & Reproducibility

- Batches YAML (templates em `examples/batches/`) para experimentos repetíveis.

- Configuração via `.env` e `config/settings.yaml`; diretórios de datasets/outputs criados automaticamente.

- Seeds determinísticos quando informados; resultados com esquema padronizado para comparação lado a lado.

Conclusion

Backed by a modular architecture and practical baseline, heuristic, and exact methods, CSP-Bench enables reproducible, apples-to-apples comparisons across datasets and parameterizations. It helps quantify trade-offs between quality and runtime and provides a clear path to integrate future algorithms. The broader context from heuristic advances (Abdi et al., 2024) and complexity limits (Abboud et al., 2023) reinforces the value of a dedicated benchmarking toolkit.

Acknowledgements

The author thanks project contributors, users, and the broader computational biology community for feedback and discussions that helped shape the framework. No external financial support was received for the development of this software.

Abboud, A., Fischer, N., Goldenberg, E., S., K. C., & Safier, R. (2023). Can you solve closest string faster than exhaustive search? *31st Annual European Symposium on Algorithms (ESA 2023)*, 3:1–3:17. https://doi.org/10.4230/LIPIcs.ESA.2023.3

Abdi, A., Djukanovic, M., Tahmasebi Boldaji, H., Salehi, H., & Kartelj, A. (2024). A three-stage algorithm for the closest string problem on artificial and real gene sequences. *arXiv Preprint*. https://arxiv.org/abs/2407.13023