The Journal of Open Source Software

gdock: Information-driven protein-protein docking using a genetic algorithm

Rodrigo V. Honorato [1¶]

1 Computational Structural Biology Group, Utrecht University, The Netherlands ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Proteins carry out most biological functions by interacting with other proteins, and understanding these interactions at the molecular level is essential for drug design and biomedical research. Computational docking predicts how two proteins bind together by searching for arrangements that are both physically plausible and consistent with experimental data.

gdock is a command-line tool that performs protein-protein docking guided by user-supplied restraints—information about which residues are likely at the binding interface. It uses a genetic algorithm to efficiently explore possible orientations of one protein relative to another, scoring each candidate with a physics-based energy function. Written entirely in Rust, `gdock` compiles to a single executable with no external dependencies, making it straightforward to install and integrate into automated workflows. On a standard workstation, most docking runs complete in under 20 seconds. Documentation and source code are available at https://github.com/rvhonorato/gdock and https://gdock.org.

Statement of need

Information-driven docking incorporates experimental data—from mutagenesis, cross-linking mass spectrometry, or NMR—to guide protein complex structure prediction (Noort et al., 2021). While several tools support this approach, they typically require complex runtime environments or offer limited restraint integration. `gdock` contributes to this ecosystem as a fast, minimal implementation:

- **Single binary**: No runtime dependencies or environment setup
- **Speed**: ~15 seconds per complex on standard hardware
- **Rust implementation**: Native energy functions, memory-safe, readable
- **CLI-first**: Designed for scripting and pipeline integration

The tool provides an accessible option for researchers who need restraint-driven docking without the overhead of larger software packages.

State of the field

Protein-protein docking software spans a range of complexity and capability. For example, ClusPro (Kozakov et al., 2017) and ZDOCK (Pierce et al., 2014) provide FFT-based sampling with web interfaces, though restraint integration is limited. HADDOCK (Dominguez et al., 2003) offers comprehensive information-driven docking with flexible refinement, symmetry handling, and multi-body support; LightDock (Jiménez-García et al., 2018) uses swarm optimization with restraint support—both require managed Python environments with specific package versions, and HADDOCK additionally depends on CNS (Crystallography and NMR System)

(Brünger et al., 1998). A limited Rust implementation of LightDock exists (Jiménez-García, 2020) and served as one inspiration for `gdock`.

`gdock` occupies a distinct niche: a dependency-free, single-binary tool for restraint-driven rigid-body docking. Rather than extending existing software— which would require adapting to their architectural constraints—`gdock` was built from scratch in Rust to prioritize minimal deployment overhead and scripting integration. Crucially, the entire scoring function is implemented from scratch in modern, readable code, making it fully transparent and easy to verify—unlike tools that depend on legacy Fortran engines or opaque external libraries. `gdock` does not aim to replace full-featured docking platforms but provides a lightweight alternative for users with reliable interface information who need rapid, reproducible results without environment setup.

Software design

`gdock` is a Rust rewrite of an earlier Python prototype, compiling to a ~7,000-line statically-linked binary with no runtime dependencies.

Search algorithm. A genetic algorithm explores rigid-body transformations of the ligand relative to the receptor. Each chromosome encodes six genes: three Euler angles (α, β, γ) for rotation and three displacement values (x, y, z) for translation. A generation consists of a population of chromosomes that evolves through tournament selection, uniform crossover, creep mutation (Gaussian perturbations for local refinement), and elitism. Fitness evaluation is parallelized across the population. The search terminates early upon convergence.

Scoring function. The energy function combines four terms:

$$E_{total} = w_{vdw}E_{vdw} + w_{elec}E_{elec} + w_{desolv}E_{desolv} + w_{air}E_{air}$$

- E_{vdw}: Soft-core Lennard-Jones potential that remains finite at short distances, allowing the search to explore conformations with minor clashes
- E_{elec}: Coulombic interactions with distance-dependent dielectric ($\varepsilon = r$) to dampen long-range effects
- E_{desolv}: Empirical atomic solvation parameters penalizing burial of polar atoms and rewarding burial of hydrophobic atoms
- E_{air}: Flat-bottom harmonic potential on Cα–Cα distances between user-specified residue pairs (no penalty within 0–7 Å, quadratic penalty beyond), conceptually inspired by HADDOCK's distance restraints (Dominguez et al., 2003) but using a simpler purely harmonic form without the linear switching at long distances

Weight calibration. The weights w_{vdw}, w_{elec}, and w_{desolv} were calibrated using the Dockground decoy set (Gao et al., 2008), which provides 100 decoy structures per complex with exactly one near-native conformation. A grid search tested weight combinations by re-scoring all decoys and measuring how often the near-native structure ranked in the top 50. The final weights ($w_{vdw} = 0.4$, $w_{elec} = 0.05$, $w_{desolv} = 3.4$) maximize this ranking performance. The restraint weight w_{air} is fixed at 1.0 since calibration was performed without restraints.

Output. Final models are clustered using Fraction of Common Contacts (FCC) (Rodrigues et al., 2012), re-implemented natively in Rust, and ranked by score, providing both diverse and top-scoring solutions.

Code quality. The codebase includes 174 unit tests covering parsing, energy calculations, and algorithm behavior. Continuous integration enforces code formatting (`rustfmt`), linting (`clippy` with warnings as errors), and test passage on every commit. Rust's ownership model provides compile-time guarantees against data races and use-after-free errors. The software is released under the permissive 0BSD license.

Research impact statement

gdock was validated on 271 complexes from the Protein-Protein Docking Benchmark v5 (Vreven et al., 2015), a standard dataset for assessing methods, using the bound-bound conformations. Restraints were derived as explicit Cα–Cα residue pairs from native interface contacts (within 5 Å), simulating ideal contact information. Unlike HADDOCK's ambiguous interface restraints, each restraint specifies a single receptor–ligand residue pair.

Using the DockQ metric (Basu & Wallner, 2016) to assess model quality, gdock achieved a 95.9% success rate (260/271 complexes with at least one acceptable model, DockQ 0.23). Medium-quality models (DockQ 0.49) were obtained for 55.7% of complexes, and high-quality models (DockQ 0.80) for 3.7% (Figure 1).

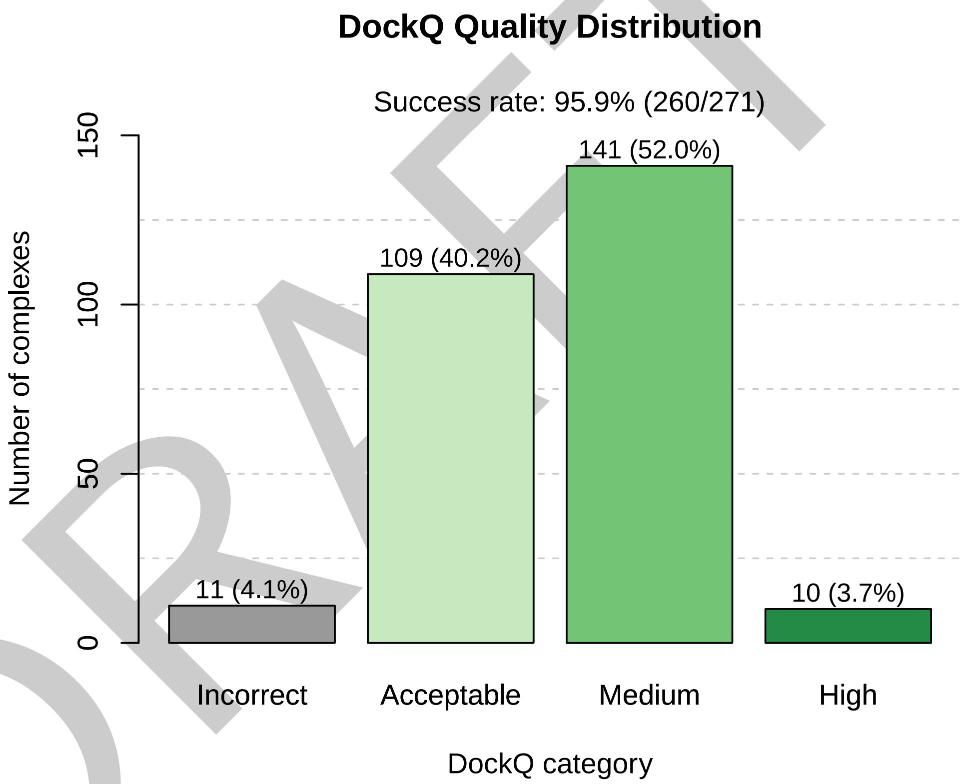

Figure 1: Distribution of docking quality across 271 benchmark complexes. Each complex is categorized by its best DockQ score among 10 output models.

Performance benchmarks on a 48-core machine show a median docking time of ~15 seconds per complex, with 56% of cases completing within 20 seconds (Figure 2).

The Journal of Open Source Software

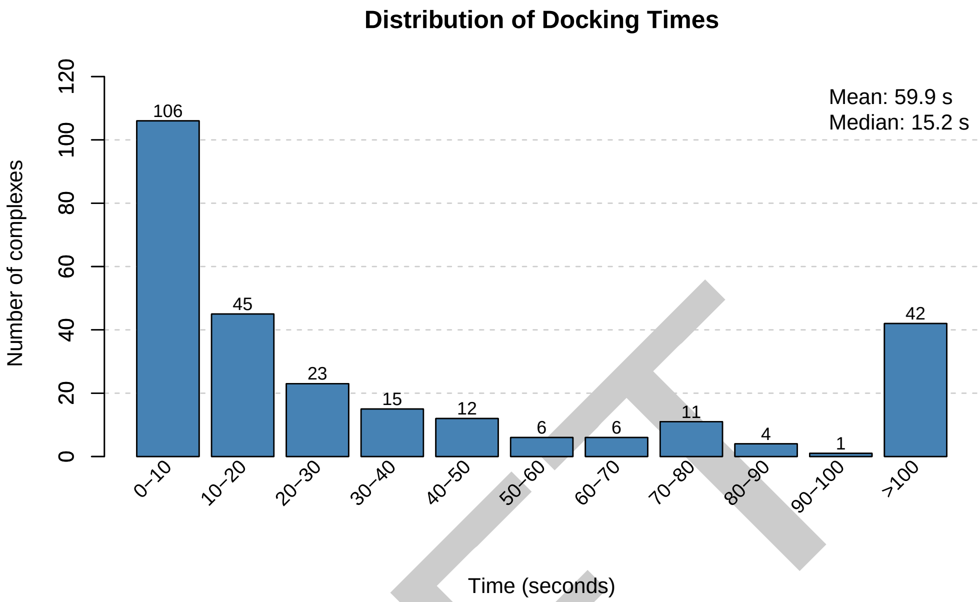

Figure 2: Distribution of docking times across benchmark complexes. Most cases complete within 20 seconds; outliers correspond to larger protein systems.

These results reflect ideal restraint conditions; real-world performance depends on restraint quality. As a rigid-body method, gdock is best suited for cases where conformational changes upon binding are minimal.

Scripts to reproduce these experiments are available at: https://github.com/rvhonorato/gdock-benchmark.

AI usage disclosure

Claude (Anthropic) assisted with code review, test generation, and proofreading. All AI-generated content was verified by the author through manual review and the continuous integration pipeline.

Acknowledgements

The author thanks Prof. Dr. Alexandre Bonvin for computational resources and expertise, and Dr. Brian Jiménez-García for early conceptualization.

References

Basu, S., & Wallner, B. (2016). DockQ: A quality measure for protein-protein docking models. *PloS One*, *11*(8), e0161879. https://doi.org/10.1371/journal.pone.0161879

Brünger, A. T., Adams, P. D., Clore, G. M., DeLano, W. L., Gros, P., Grosse-Kunstleve, R. W., Jiang, J.-S., Kuszewski, J., Nilges, M., Pannu, N. S., Read, R. J., Rice, L. M., Simonson, T., & Warren, G. L. (1998). Crystallography & NMR system: A new software suite for macromolecular structure determination. *Acta Crystallographica Section D: Biological Crystallography*, *54*(5), 905–921. https://doi.org/10.1107/S0907444998003254

Dominguez, C., Boelens, R., & Bonvin, A. M. (2003). HADDOCK: A protein-protein docking approach based on biochemical or biophysical information. *Journal of the American Chemical Society*, *125*(7), 1731–1737. https://doi.org/10.1021/ja026939x

Gao, Y., Douguet, D., Tovchigrechko, A., & Vakser, I. A. (2008). DOCKGROUND protein-protein docking decoy set. *Bioinformatics*, *24*(22), 2634–2635. https://doi.org/10.1093/bioinformatics/btn497

Jiménez-García, B. (2020). *Lightdock-rust: Rust implementation of the LightDock macromolecular docking framework*. https://github.com/lightdock/lightdock-rust

Jiménez-García, B., Roel-Touris, J., Romero-Durana, M., Vidal, M., Jiménez-González, D., & Fernández-Recio, J. (2018). LightDock: A new multi-scale approach to protein-protein docking. *Bioinformatics*, *34*(1), 49–55. https://doi.org/10.1093/bioinformatics/btx555

Kozakov, D., Hall, D. R., Xia, B., Porter, K. A., Padhorny, D., Yueh, C., Beglov, D., & Vajda, S. (2017). The ClusPro web server for protein-protein docking. *Nature Protocols*, *12*(2), 255–278. https://doi.org/10.1038/nprot.2016.169

Noort, C. W. van, Honorato, R. V., & Bonvin, A. M. (2021). Information-driven modeling of biomolecular complexes. *Current Opinion in Structural Biology*, *70*, 70–77. https://doi.org/10.1016/j.sbi.2021.05.003

Pierce, B. G., Wiehe, K., Hwang, H., Kim, B.-H., Vreven, T., & Weng, Z. (2014). ZDOCK server: Interactive docking prediction of protein-protein complexes and symmetric multimers. *Bioinformatics*, *30*(12), 1771–1773. https://doi.org/10.1093/bioinformatics/btu097

Rodrigues, J. P., Trellet, M., Schmitz, C., Kastritis, P., Karaca, E., Melquiond, A. S., & Bonvin, A. M. (2012). Clustering biomolecular complexes by residue contacts similarity. *Proteins: Structure, Function, and Bioinformatics*, *80*(7), 1810–1817. https://doi.org/10.1002/prot.24078

Vreven, T., Moal, I. H., Vangone, A., Pierce, B. G., Kastritis, P. L., Torchala, M., Chaleil, R., Jimenez-Garcia, B., Bates, P. A., Fernandez-Recio, J., & others. (2015). Updates to the integrated protein-protein interaction benchmarks: Docking benchmark version 5 and affinity benchmark version 2. *Journal of Molecular Biology*, *427*(19), 3031–3041. https://doi.org/10.1016/j.jmb.2015.07.016