RustSASA: A Rust Crate for Accelerated Solvent Accessible Surface Area Calculations

Maxwell J. Campbell [1]

1 University of California, San Francisco, United States

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Solvent accessible surface area (SASA) calculations are fundamental for understanding protein structure, function, and dynamics in computational biology. These calculations quantify the surface area of biomolecules accessible to solvent molecules, providing insights into protein folding, stability, and intermolecular interactions. The Shrake-Rupley algorithm has served as the standard for SASA calculations since 1973, but existing implementations often become computational bottlenecks when analyzing large protein datasets. As proteomics datasets continue to grow with initiatives like AlphaFold producing hundreds of millions of predicted protein structures, the need for efficient SASA calculation tools has increased dramatically. RustSASA addresses this challenge by providing a high-performance implementation of the Shrake-Rupley algorithm written in pure Rust, delivering a 5× speed improvement over FreeSASA while maintaining calculation accuracy and providing interfaces for multiple programming languages and frameworks (i.e. MDAnalysis). The source code is freely available at https://github.com/maxall41/RustSASA.

Statement of need

Current SASA calculation tools represent a significant computational bottleneck in structural biology workflows, particularly for molecular dynamics simulations and high-throughput analyses. Popular implementations such as those in Biopython and FreeSASA, while accurate, become prohibitively slow when processing large protein datasets. RustSASA addresses this performance gap by leveraging Rust's efficient parallelization abstractions (via Rayon) and readily available SIMD instructions (via Pulp). These optimizations enable RustSASA's performance advantage over the simpler C implementation of the same algorithm in FreeSASA.

Benchmarking on representative protein structures demonstrates that RustSASA achieves a 5× improvement over FreeSASA, and a 63× performance improvement over Biopython. This performance advantage reduces computational costs for high-throughput structural analyses and makes large-scale comparative studies feasible. Furthermore, RustSASA's multi-language support (Rust and Python), command-line interface, and MDAnalysis package (Gowers et al., 2016; Michaud-Agrawal et al., 2011) ensure broad accessibility across the computational biology community.

Results

Calculation Quality

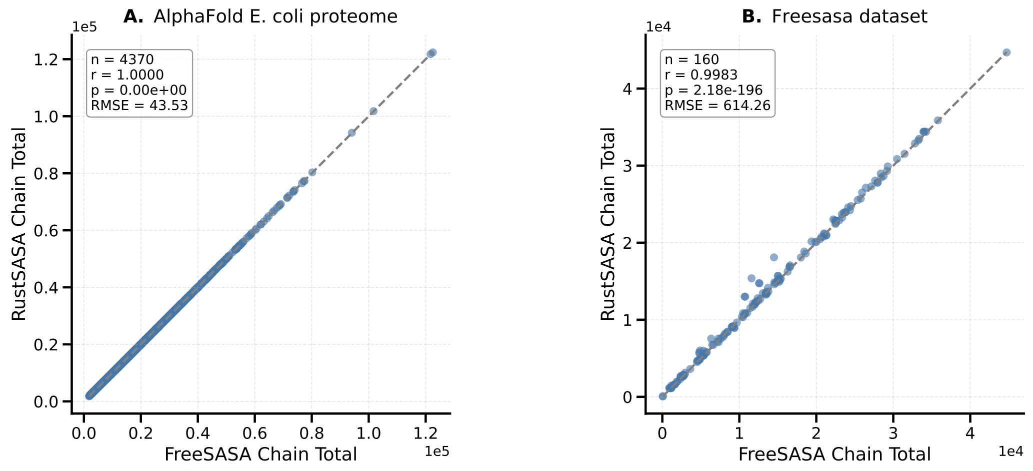

Figure 1: A. Comparing RustSASA against FreeSASA on *E. coli* proteome at the chain level. **B.** Comparing RustSASA against FreeSASA on FreeSASA comparison dataset at the chain level.

To evaluate the accuracy of RustSASA calculations, we compared results to FreeSASA (Mitternacht, 2016) on both the predicted *E. coli* proteome from AlphaFold DB (Jumper et al., 2021; Varadi et al., 2021) and the FreeSASA evaluation dataset. RustSASA produces SASA values that closely match those from FreeSASA, achieving Pearson correlation coefficients $>$ 0.99 on both datasets (Figure 1).

Performance

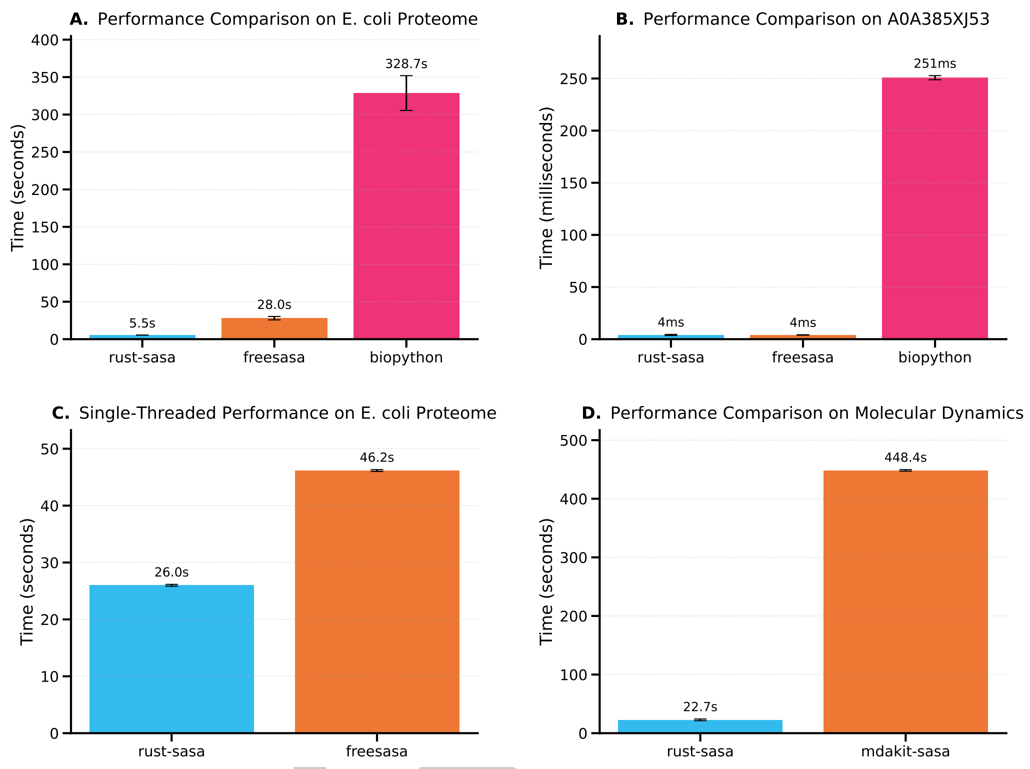

Figure 2: A. Comparing the multi-threaded performance of RustSASA, FreeSASA, and Biopython on the full AlphaFold *E. coli* proteome. **B.** Comparing the multi-threaded performance of RustSASA, FreeSASA, and Biopython on A0A385XJ53, a protein randomly selected from the AlphaFold *E. coli* proteome. **C.** Comparing the single-threaded performance of RustSASA and FreeSASA on the full AlphaFold *E. coli* proteome. **D.** Comparing the performance of RustSASA and mdakit-sasa (based on freesasa) on a molecular dynamics trajectory

We evaluated the performance of FreeSASA, RustSASA, and Biopython (Cock et al., 2009) across four evaluations. First, we performed multi-threaded SASA calculations for all proteins in the *E. coli* proteome. Second, we evaluated the performance of these methods on a single randomly selected protein (A0A385XJ53) from the AlphaFold *E. coli* proteome. Third, we evaluated the single-threaded performance of RustSASA and FreeSASA on the *E. coli* proteome, Biopython was excluded from this benchmark due to its poor performance hindering timely evaluation. Fourth, we evaluated the performance of RustSASA and mdakit-sasa, which uses freesasa internally, on a molecular dynamics trajectory, specifically trajectory 10824 (4IAQ, 5HT receptor) from GPRCmd (Rodríguez-Espigares et al., 2020).

For the full proteome benchmarks (Figure 2A), we used Hyperfine (Peter, 2023) with 3 runs and 3 warmup iterations. All methods utilized parallel processing across eight cores. GNU parallel (Tange, 2011) was used to parallelize FreeSASA and Biopython, while RustSASA utilized its internal parallelization. RustSASA processed the entire proteome in ~5 seconds compared to ~28 seconds for FreeSASA and ~328 seconds for Biopython, representing 5× and 63× speed improvements, respectively.

For the single protein benchmark (Figure 2B), we used Hyperfine with 3 warmup iterations and 25 runs. RustSASA processed the protein in 4.0ms (±0.5), FreeSASA processed the protein in 4.0ms (±0.2), and Biopython processed the protein in 250.8ms (±2.0). On the single-threaded benchmark (Figure 2C), RustSASA processed the proteome in 26.0 seconds compared to 46.2 seconds for FreeSASA, representing a ~43% performance improvement, demonstrating that

RustSASA's performance advantage is not solely due to multi-threading.

For the molecular dynamics trajectory benchmark (Figure 2D), we used Hyperfine with 3 runs. RustSASA processed the trajectory in 22.7 seconds (±1.4), and mdakit-sasa processed the trajectory in 448.4 seconds (±1.3), representing a ~20x performance improvement. The magnitude of the improvement can be attributed to the inefficiency with which mdakit-sasa utilizes FreeSASA.

Methods

RustSASA computes solvent-accessible surface areas (SASA) using the Shrake–Rupley algorithm (Shrake & Rupley, 1973). In this algorithm each atom is represented as a sphere with a radius equal to its atomic van der Waals radius plus the radius of a spherical solvent probe; the sphere surface is sampled with a dense quasi-uniform distribution of test points and a point is considered solvent accessible if it is not occluded by any neighboring atom sphere. For all calculations reported here a solvent probe radius of 1.4 Å (the approximate radius of a water molecule) was used.

Atomic radii were assigned according to the ProtOr parameter set introduced by Tsai et al. (Tsai et al., 1999). These radii were applied to all non-hydrogen heavy atoms present in the structures; hydrogen atoms, when present in input files, were ignored for the SASA computations to maintain consistency with common practice for protein SASA estimation. Additionally, all HETATM records in the input—non-standard amino acids and ligands—were ignored.

To ensure a fair comparison of RustSASA and FreeSASA in the single-threaded benchmark, a C++ script was utilized to call the FreeSASA C API for all input proteins in a given folder. This approach ensures that the command-line overhead is not responsible for RustSASA's performance advantage. Furthermore, in all experiments, FreeSASA was configured to use the Shrake-Rupley algorithm over its default algorithm, Lee & Richards, to ensure an accurate comparison between the methods. Proteome-scale structure models for Escherichia coli were obtained from the AlphaFold DB (entry UP000000625_83333_ECOLI_v6, available at https://alphafold.ebi.ac.uk/download). All experiments were conducted on a 2024 Apple MacBook Air with an M3 processor and 24GB of unified memory.

Conclusion

RustSASA provides a significant advancement in SASA calculation performance while maintaining accuracy, addressing a bottleneck in computational structural biology. The 5× speed improvement over current standards enables previously intractable analyses of large protein datasets and molecular dynamics simulations. By providing interfaces for multiple programming languages alongside a command-line tool and MDAnalysis package, RustSASA ensures broad accessibility across the research community. As structural biology datasets continue to expand, efficient computational tools like RustSASA become essential for advancing our understanding of protein structure and function.

Acknowledgements

We would like to thank Rodrigo Honorato and Niccolò Bruciaferri for their valuable contributions to this project. We would also like to thank the reviewers for their insightful comments.

References

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & others. (2009). Biopython: Freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

Gowers, R. J., Linke, M., Barnoud, J., Reddy, T. J. E., Melo, M. N., Seyler, S. L., Domański, J., Dotson, D. L., Buchoux, S., Kenney, I. M., & Beckstein, O. (2016). MDAnalysis: A python package for the rapid analysis of molecular dynamics simulations. *SciPy 2016*. https://doi.org/10.25080/Majora-629e541a-00e

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry*, *32*(10), 2319–2327. https://doi.org/https://doi.org/10.1002/jcc.21787

Mitternacht, S. (2016). FreeSASA: An open source c library for solvent accessible surface area calculations. *F1000Research*, *5*, 189. https://doi.org/10.12688/f1000research.7931.1

Peter, D. (2023). *Hyperfine*. https://github.com/sharkdp/hyperfine

Rodríguez-Espigares, I., Torrens-Fontanals, M., Tiemann, J. K. S., Aranda-García, D., Ramírez-Anguita, J. M., Stepniewski, T. M., Worp, N., Varela-Rial, A., Morales-Pastor, A., Medel-Lacruz, B., Pándy-Szekeres, G., Mayol, E., Giorgino, T., Carlsson, J., Deupi, X., Filipek, S., Filizola, M., Gómez-Tamayo, J. C., Gonzalez, A., … Selent, J. (2020). GPCRmd uncovers the dynamics of the 3D-GPCRome. *Nature Methods*, *17*(8), 777–787. https://doi.org/10.1038/s41592-020-0884-y

Shrake, A., & Rupley, J. A. (1973). Environment and exposure to solvent of protein atoms. Lysozyme and insulin. *Journal of Molecular Biology*, *79*(2), 351–371. https://doi.org/https://doi.org/10.1016/0022-2836(73)90011-9

Tange, O. (2011). GNU parallel - the command-line power tool*;Login: The USENIX Magazine*, *36*(1), 42–47. http://www.gnu.org/s/parallel

Tsai, J., Taylor, R., Chothia, C., & Gerstein, M. (1999). The packing density in proteins: Standard radii and volumes11Edited by j. M. thornton. *Journal of Molecular Biology*, *290*(1), 253–266. https://doi.org/https://doi.org/10.1006/jmbi.1999.2829

Varadi, M., Anyango, S., Deshpande, M., Nair, S., Natassia, C., Yordanova, G., Yuan, D., Stroe, O., Wood, G., Laydon, A., Žídek, A., Green, T., Tunyasuvunakool, K., Petersen, S., Jumper, J., Clancy, E., Green, R., Vora, A., Lutfi, M., … Velankar, S. (2021). AlphaFold protein structure database: Massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, *50*(D1), D439–D444. https://doi.org/10.1093/nar/gkab1061