Probando la eficiencia de OpenMdlr, 2024, 1–5
Advance Access Publication Date: 04-4-2024
Paper

Enhancing Protein Modeling: The Parallelization of OpenMDlr

Fernández Costas, Xoán[1,*]

Abstract

OpenMDlr, an innovative program written in Python, in conjunction with AmberTools, offers users a straightforward and user-friendly platform for conducting protein folding predictions. In this study, our objective is to explore the efficacy of program parallelization by analyzing three chosen proteins: 2ERL, 2LYZ, and 1UBQ. Additionally, we aim to assess the program's ability to predict protein structures relative to their crystalline forms. Our analysis will encompass various aspects, ranging from scalability to the accuracy of OpenMDlr predictions.

Key words: Protein modeling, Parallelization, Structural biology, Computational efficiency.

1. Introduction

Parallelization has become a fundamental pillar in the field of computing, enabling efficient processing of large volumes of data and the simultaneous execution of complex tasks. The development of parallel programs has experienced significant growth, driven by the need to fully utilize available computational resources and enhance application performance.

Modeling protein structure using distance constraints is a common task in structural biology. However, protein modeling and simulation are computationally expensive processes that require high processing power and an efficient approach to reduce execution times. Structure prediction from sequence has recently seen significant advances, and even methods that directly predict atomic coordinates make use of some form of constraint-based refinement (Jumper et al., 2021). Until 2022, there was a notable absence of a lightweight, general-purpose, user-friendly, open-source program designed for rapidly modeling protein structures using contemporary physical potentials and pairwise interatomic distance constraints. Recognizing this gap, Davidson and colleagues introduced OpenMDlr (Davidson et al., 2022), an open-source tool that harnesses modern biomolecular force fields and the AmberTools simulation engine within a user-friendly framework. OpenMDlr is specifically tailored for modeling protein structures from arbitrary sets of pairwise atomic distances. Figure 1 illustrates the modeling procedure.

This study focuses on evaluating the efficiency of OpenMDlr parallelization by analyzing its ability to scale with an increasing number of processing cores and structurally analyzing the protein foldings generated by the program.

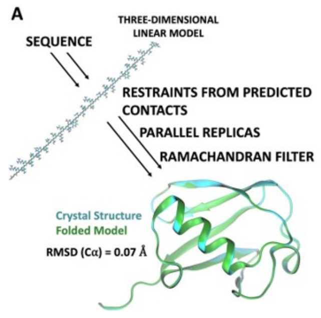

Fig. 1: Modeling procedure in OpenMDlr. A sequence and a set of distances is input into the program, which generates a linear peptide from the sequence, then runs parallel restrained molecular dynamics simulations to produce a large set of structures (Davidson et al., 2022).

2. System and methods

The FinisTerrae III provides a high-performance computing infrastructure intended for analyzing the time and accuracy of protein structure predictions during program execution. This supercomputer boasts a total computing power of 4.36 PetaFLOPS, comprising 357 nodes interconnected via an HDR Infiniband network. Additionally, it features the latest generation Intel Xeon Ice Lake 8352Y processors with 32 cores at 2.2GHz (22,848 cores) and 157 GPUs (141 Nvidia A100 and 16 Nvidia T4), making it suitable for running OpenMDlr.

2.1. OpenMDlr setup

OpenMDlr utilizes the MIT license, making it a fully permissive open-source program. The dependencies it uses are also open-source and limited to MDAnalysis (GNU General Public License), AmberTools (GNU General Public License), Joblib (BSD license), NumPy (modified BSD license), and Biopython (Biopython license).

The OpenMDlr program consists of three main phases:

1. Analysis and preprocessing of data: In this initial stage, the input data, including primary protein sequences and structural constraints, are analyzed and processed.
2. Modeling and molecular simulation: OpenMDlr utilizes advanced molecular modeling algorithms and dynamic simulation to generate models of protein structures. To enhance the performance of this phase, MD simulation replicas are regenerated in parallel using Python's Joblib library. The replica simulations are launched simultaneously across multiple threads with the maximum number of threads defined in the input configuration file.
3. Analysis and Validation of Results: In this final step, simulation data is compared with experimental data to verify the stability of proteins and the accuracy of predicted structure.

For its basic operation, OpenMDlr necessitates minimal input, primarily a configuration file. Users are required to edit this configuration file to specify the files pertinent to the proteins they wish to generate three-dimensional predictions for. OpenMDlr operates within the Python programming environment and is executed via command line as a Python3 script.

2.1.1. Inputs

Regarding program inputs, three target proteins were chosen, and their corresponding configuration files are accessible in the author's GitHub repository (https://github.com/BSDExabio/OpenMDlr-amber). To commence the program, only the configuration file is necessary, wherein various paths are specified to facilitate the generation of highly accurate folding. Within this configuration file, users must specify the location of the FASTA file containing the sequence of the target protein, as well as additional files containing the prescribed ranges for interatomic or peptide dihedral distances. Furthermore, the user is prompted to determine the number of simulation replicas for the protein under investigation. Additionally, the user selects the number of simulation replicas for the protein of interest. These files can be found in the Protein Data Bank (PDB) entry of the chosen protein. The proteins chosen as inputs for the program were:

- 2ERL: Pheromone ER-1 from (Anderson, Weiss & Eisenberg, 1996).
- 2LYZ: Real-space refinement of the structure of hen egg-white lysozyme (Diamond, 1974).
- 1UBQ: Structure of ubiquitin refined at 1.8 angstroms resolution (Vijay-Kumar, Bugg & Cook, 1987).

2.1.2. Outputs

Among the various program outputs, there are main files shared across all simulations, including those created and generated by TLEAP, a component of the Amber software package for molecular simulation. These files contain the representation of all atoms in the three-dimensional structure of the protein (fully folded), which will be used to initiate the folding simulations. Additionally, there are other files displaying different statistics for each atom and the time required by the program, which will serve to measure its parallel time.

In each independent execution of the folding simulation, there is a final folded structure (with .pdb extension), which we will represent and observe using the VMD (Visual Molecular Dynamics) tool. VMD allows visualization and analysis of molecular structures (Humphrey, Dalke, & Schulten, 1996).

2.1.3. Parallelization

OpenMDlr utilizes the sander software from AmberTools21 in single-core mode to perform molecular dynamics simulations. OpenMDlr employs a strategy of perfectly distributed parallel workload implementation to efficiently execute replicas across the available cores, as defined within the configuration file. It's worth noting that the total completion time depends on the number of replicas conducted as well as the number of available cores for executing these replicas.

2.2. Performance of OpenMDlr

In assessing the parallelization efficiency of OpenMDlr on the FinisTerrae III, multiple distinct configurations were employed for each of the chosen proteins. Specifically, the program's ability to maintain optimal performance and functionality as the number of cores associated with the node where the program was running increased was measured. In order to assess scalability, the program was run on a single node while gradually increasing the number of cores. Consequently, program execution times were recorded for 1, 2, 4, 8, 16, 32, and 64 cores.

Nonetheless, it is advisable to select an optimal configuration and design to guarantee reliable scalability. Therefore, separate independent simulations were conducted for each of the proteins:

- Folding of protein 2ERL with 20 simulations.
- Folding of protein 2LYZ with 15 simulations.
- Folding of protein 1UBQ with 64 simulations.

To represent the scalability of the program, metrics such as speedup were calculated, represented by the formula $S(p) = \frac{T_{\text{sec}}}{T_{\text{par}}(p)}$, which indicates the improvement in performance when using multiple cores. Also calculated was the Overhead, defined as $\text{Overhead} = C(p) - T_{\text{sec}}$, which shows the difference between parallel execution time and sequential execution time. Finally, efficiency was used, expressed by $E(p) = \frac{T_{\text{sec}}}{\text{Cost}(p)}$, to assess the additional processing power when increasing the number of processors in a parallel system.

2.3. TM-Score

To verify the accuracy of the three-dimensional folding of each protein generated by OpenMDlr, the TM-Score was used. TM-Score is a metric that quantifies the structural similarity between two protein structures by aligning corresponding residues and calculating a score indicating how much the structures overlap (Zhang & Skolnick, 2004). The TM-Score executable program was downloaded from: http://bioinformatics.buffalo.edu/TM-score.

3. Results and Discussion

This section furnishes an overview of the study's findings, facilitating a lucid and concise comprehension of the results and their significance within a broader context.

3.1. Scalability

Table 1 illustrates the outcomes derived from executing the program with the three proteins under investigation (2ERL, 2LYZ, and 1UBQ). The table delineates the utilization of OpenMDlr across diverse configurations on a single node equipped with multiple cores. Across all instances, notable enhancements in efficiency and substantial reductions in program duration were evident compared to sequential execution. Specifically, execution times remained below 8 minutes for all cases when employing 64 cores.

In the first case, with the folding of protein 2ERL using 20 simulations, the time is halved in the first 4 nodes until it stabilizes around 32n, reducing speedup and reaching an optimal time of approximately 1 minute and 13 seconds. For the second folding involving protein 2LYZ and employing 15 simulations, it's pertinent to highlight that it represents the largest protein among the three selected. The sequential program necessitates nearly 2 hours for completion. Continuing with Table 1, a noteworthy reduction in processing time is evident starting from 16 threads. After surpassing this inflection point, the execution time stabilizes around 6 minutes and 50 seconds, also resulting in a lesser speedup compared to the case of 2ERL, attributed to the higher complexity associated with this protein. Finally, for 1UBQ with 64 simulations, as we increase the number of nodes, the time is halved in each of the tests.

Other metrics were also calculated to gauge the efficiency of parallelization, such as overhead and efficiency. Concerning overhead, in the first two cases (2ERL and 2LYZ), the time increases as the number of cores rises, indicating additional computational load in our parallelization. Specifically, with 64 cores, an overhead of 53 minutes is observed for 2ERL and 330 minutes for 2LYZ. While the efficiency remained above 90% in both cases up to 16 cores, it subsequently declined from 92% to 30% and from 94% to 24%, respectively. This suggests a minor limitation in the program's scalability, particularly in the configuration of the 2LYZ protein.
However, concerning the 1UBQ protein, a peak overhead of 18 minutes is observed at the second core, which subsequently decreases to 2 minutes at the fourth core, before rising again to 43 minutes at the 64th core. Nonetheless, despite these fluctuations, the program's efficiency remains high, exceeding 80% throughout, in contrast to the first two proteins. In fact, it even reaches 83% at the 64th core, indicating robust scalability for this configuration.

In Figure 2A, the scalability of each protein configuration is depicted. The vertical axis represents speedup, while the horizontal axis displays the number of cores used for simulations. It is evident that the folding of the 2ERL protein demonstrates improved scalability up to 16 nodes. However, beyond this point, the program's efficiency plateaus. Similarly, with 2LYZ, scalability increases until reaching 16 nodes, after which the speedup does not show further improvement. However, the best results are achieved with large simulation replicas, as seen in the case of 1UBQ (with 64 replicas), where we observe linear and highly efficient scalability up to 64 cores.
On the contrary, in Figure 2B, the vertical axis depicts time in minutes. This representation showcases the efficiency of parallelization, as evident from the significant reduction in total execution time to less than 8 minutes across all cases. Notably, the folding of 1UBQ, which previously consumed 3 hours, is now completed within this shortened timeframe.

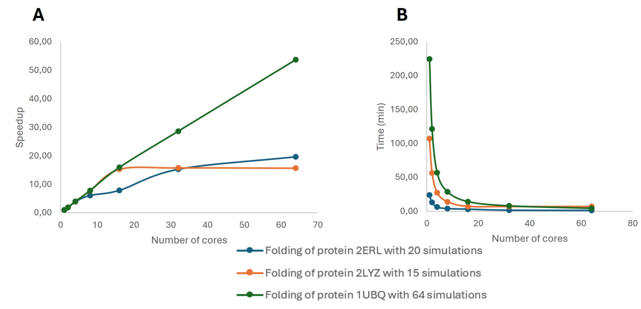

Fig. 2: Scalability Representation. A) Speedup as a function of the number of processing cores. B) Execution time (in minutes) as a function of the number of processing cores.

The parallelization of the program up to 64 nodes has revealed opportunities for improvement. In the first two cases, the speedup stabilizes, while the overhead increases and efficiency decreases. However, for the 1UBQ case, favorable results are observed in the parallelization of the program. This is mainly attributed to its design, as demonstrated by Davinson et al. (2022), who indicates that the use of 160 replicas improves the results compared to 10 or 64 replicas. Generally, it is suggested to use more replicas to achieve better outcomes.

3.2. Protein Folding

It was proposed to conduct an analysis after the execution of the different proteins to assess how efficient the folding results

Table 1. Measurement of the scalability of 2ERL, 2LYZ, and 1UBQ with 20, 15, and 64 simulations respectively in terms of execution time and as a function of the number of cores. Results are shown for configurations of 1, 2, 4, 8, 16, 32, and 64 cores.

Configuration	Execution Time						
	Number of Cores						
	1n	2n	4n	8n	16n	32n	64n
Folding of protein 2ERL	0:23:57	0:12:57	0:06:09	0:03:56	0:03:02	0:01:34	0:01:13
Folding of protein 2LYZ	1:46:59	0:56:34	0:27:14	0:13:55	0:07:01	0:06:48	0:06:50
Folding of protein 1UBQ	3:44:35	2:01:25	0:56:45	0:28:45	0:14:06	0:07:51	0:04:11

are compared to the protein's real structure. Therefore, the TM-Score was used as a measure to compare structural similarity by comparing the crystal structure of the protein with each of the simulated proteins:

- 2ERL: In Figure 3A, the crystal structure is represented in purple, while the folding result of the protein is depicted in green. We can observe that this structure contains 3 alpha helices in its secondary structure. Regarding the TM-Score, a value of 0.9264 was obtained. This indicates a fairly high similarity between the three-dimensional conformations, suggesting that they are very similar.

- 2LYZ: This protein is the one with the most amino acids that has been folded. In image 3B, we can again observe the crystal structure (in purple) and the structure predicted by the program (in green). Furthermore, a TM-Score of 0.9850 was obtained, indicating that the structure predicted by the program is very similar to the original protein.

- 1UBQ: Ubiquitin is a protein of great relevance due to the biological function it performs, so modeling its structure can be a vital process. In Figure 3C, the conformation of its crystal structure is compared to the structure predicted by OpenMDlr. On this occasion, a TM-Score of 0.9684 was obtained, indicating that it is the same structure.

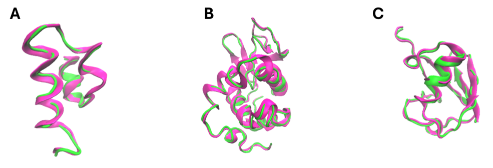

Fig. 3: Structural Comparison Between Crystal Structure and OpenMDlr Predictions: A) Protein 2ERL with a TM-Score of 0.9264. B) Protein 2LYZ with a TM-Score of 0.9850. C) Protein 1UBQ with a TM-Score of 0.9684. Purple denotes the crystal structure, while green signifies the structural prediction using OpenMDlr.

4. Conclusions

Based on the results derived from parallelizing OpenMDlr, it is evident that optimal execution is consistently achieved across all cases, resulting in a significant reduction in processing time. Coupled with the high-quality results obtained, characterized by a TM-Score exceeding 90%, the following conclusions can be drawn:

1. The parallelization of the program up to 64 nodes has demonstrated potential for improvement in all cases, with the configuration of 1UBQ being the most efficient. Despite this, a considerable reduction in time was achieved.
2. It has been observed that despite the performance, the program seems to reach a limit in execution time, where it cannot further reduce the time.
3. The high quality of the results obtained, with a TM-Score exceeding 90%, confirms the validity and accuracy of the protein folding method used by the program.

The work carried out in parallelizing OpenMDlr has proven to be promising, highlighting both its computational efficiency and the quality of the results, suggesting a significant advancement in the field of protein modeling and its potential applications in structural biology and medicine. OpenMDlr promises to be a valuable tool for research in structural biology and bioinformatics, offering new insights and possibilities in the study of proteins and their functions in biological systems.

5. Acknowledgments

This research project was made possible through the access granted by the Galician Supercomputing Center (CESGA) to its supercomputing infrastructure. The supercomputer FinisTerrae III and its permanent data storage system have been funded by the Spanish Ministry of Science and Innovation, the Galician Government and the European Regional Development Fund (ERDF).

References

Anderson, D. H., Weiss, M. S., & Eisenberg, D. (1996). A challenging case for protein crystal structure determination: the mating pheromone Er-1 from *Euplotes raikovi*. *Acta Crystallographica Section D: Biological Crystallography*, *52*(3), 469-480.

Davidson, R. B., Woods, J., Effler, T. C., Thavappiragasam, M., Mitchell, J. C., Parks, J. M., & Sedova, A. (2022). OpenMDlr: parallel, open-source tools for general protein structure modeling and refinement from pairwise distances. *Bioinformatics*, *38*(12), 3297-3298.

Diamond, R. (1974). Real-space refinement of the structure of hen egg-white lysozyme. *Journal of Molecular Biology*, *82*(3), 371-391.

Humphrey, W., Dalke, A., & Schulten, K. (1996). VMD - Visual Molecular Dynamics. *Journal of Molecular Graphics*, *14*, 33-38.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... & Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583-589.

Vijay-Kumar, S., Bugg, C. E., & Cook, W. J. (1987). Structure of ubiquitin refined at 1.8 Å resolution. *Journal of Molecular Biology*, *194*(3), 531-544.

Zhang, Y., & Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, *57*(4), 702-710.