Mathematical Problems of Computer Science 63, 81–101, 2025.
doi:10.51408/1963-0134

UDC 544.12, 004.942

FlashRMSD: An Effective Approach for Symmetry-Corrected RMSD Calculation with Extensive Benchmark Analysis

Vahagn N. Altunyan

Yerevan State University, Yerevan, Armenia
e-mail: altunyanv@gmail.com

Abstract

Root-mean-square deviation (RMSD) is a crucial metric for quantifying molecular structure similarity. However, the associated combinatorial challenges complicate the calculation process when dealing with highly symmetric molecules. Although several open-source tools have been developed to perform symmetry-corrected RMSD computations, each has limitations in terms of speed, accuracy, or usability. In this paper, we introduce FlashRMSD, a novel, rapid approach for symmetry-corrected RMSD calculation. In addition, we present an extensive benchmark dataset to evaluate RMSD calculation tools and provide a comparative analysis of existing methods alongside our proposed tool.
Keywords: Symmetry corrected RMSD, FlashRMSD, Molecular docking, Backtracking.
Article info: Received 30 March 2024; sent for review 1 April 2025; accepted 2 May 2025.

1. Introduction

Root Mean Square Deviation (RMSD) is a cornerstone metric in computational chemistry, widely employed to measure the similarity between molecular conformations. It is pivotal in applications such as assessing docking outcomes, guiding lead optimization, and filtering large sets of candidate structures in high-throughput screening. However, RMSD calculations become problematic when molecules exhibit symmetry—such as repeated functional groups or identical substituents—because standard atom-to-atom mappings often ignore these chemical equivalences. This oversight can produce inflated RMSD values and hinder accurate comparisons.

Several open-source tools attempt to address these symmetry-related challenges, but each exhibits notable constraints in terms of computational efficiency. Moreover, the field currently lacks a standardized dataset that captures the full breadth of symmetrical molecular structures. This absence complicates the fair evaluation of different RMSD methods, as it is challenging to determine whether the observed failures originate from the algorithms themselves or from insufficient testing.

Our previous studies [1] demonstrated that while existing RMSD tools can effectively process highly symmetrical structures, they often struggle with certain specific molecular configurations that are overlooked during benchmarking. These structural cases, left unexamined in typical tool evaluations, highlight gaps in current methodologies and the need for more comprehensive benchmarking datasets.

To address these challenges, we make two key contributions in this work:

1. **Comprehensive Dataset** – We curate a dataset designed to challenge RMSD tools by incorporating molecules with diverse and tricky symmetry patterns that can mislead certain tools into unnecessary computations. By spanning a broad range of molecular scaffolds, this dataset provides a rigorous benchmark for evaluating both existing and novel methods.
2. **FlashRMSD: A Symmetry-Corrected RMSD Tool** – We introduce ***FlashRMSD***, an efficient approach for symmetry-aware RMSD calculation. Our method leverages an optimized backtracking algorithm with pruning strategies to account for chemical equivalences, ensuring both accuracy and computational efficiency.

The remainder of this paper is structured as follows. First, in Section 2, we describe the construction and scope of our new dataset. Section 3 then introduces the ***FlashRMSD*** tool, detailing its theoretical background and practical implementation. Next, in Section 4, we outline the benchmark setup used to evaluate ***FlashRMSD*** alongside other RMSD calculation tools. Finally, Section 5 presents our comparative results, and Section 6 discusses edge cases of molecules that are challenging for some or all RMSD calculation tools.

1.1. Background and Related Work

1.1.1 RMSD and Symmetry Challenges

RMSD quantifies the structural similarity between two molecular conformations by measuring the root mean squared distance between corresponding atoms. While seemingly straightforward, RMSD calculations can be undermined by molecular symmetry. In symmetrical molecules, multiple valid atom mappings exist, and failing to account for all chemically equivalent correspondences can lead to erroneous or inflated RMSD values. These inaccuracies can influence the results of tasks like molecular docking, virtual screening, and structure-based drug design, where having reliable similarity metrics is crucial.

1.1.2 Existing RMSD Tools

Several RMSD tools have been developed, each tackling different aspects of the problem with varying degrees of effectiveness:

- ***spyRMSD*[2]:**
  Designed for flexibility and ease of use, ***spyRMSD*** integrates with popular libraries such as RDKit and Open Babel, leveraging Python for accessibility. However, its reliance on libraries for graph isomorphism calculations lacks problem-specific optimizations, making it highly inefficient. Additionally, it offers limited support for bond-type variations.
- **DockRMSD[3]:**
  Optimized for computational efficiency, ***DockRMSD*** is implemented in C, allowing for rapid calculations with minimal overhead. However, its functionality is restricted to specific MOL2 file formats, and it may fail silently (e.g., via segmentation faults) when encountering format inconsistencies or complex symmetries. While it does account for bond types, it silently ignores them if no valid mappings are found.
- ***obrms*:**
  As part of the OpenBabel[4] cheminformatics toolkit, ***obrms*** supports multiple file formats and cross-RMSD calculations. While it is both efficient and versatile, its packaging introduces some overhead, making it slightly less efficient than ***DockRMSD***.

Collectively, these tools highlight a common limitation: while each addresses specific user needs, none effectively balances speed, reliability, and robust handling of symmetrical equivalences. Furthermore, the absence of a comprehensive, standardized dataset encompassing diverse symmetrical structures makes it challenging to objectively evaluate their strengths and weaknesses.

1.1.3 Motivating a New Dataset

In the absence of a dedicated dataset that systematically tests RMSD performance on symmetrical structures, evaluations often rely on ad hoc collections of molecules or focus on only a few specific chemotypes. This approach fails to capture the breadth of symmetry types encountered in real-world applications, ranging from simple ring systems to large, multiply substituted scaffolds.
By presenting a new dataset that features a wide range of symmetrical patterns, we aim to provide a benchmark that can reveal subtle performance gaps in existing RMSD tools. This resource will also serve as the testing ground for our proposed ***FlashRMSD*** tool, enabling transparent comparisons and guiding future improvements in symmetry-corrected RMSD algorithms.

2. Dataset

Our dataset was constructed using molecules from two primary sources: the **Chemical Component Dictionary (CCD)[1][5]** and the **Biologically Interesting Molecule Reference**

[1] https://www.wwpdb.org/data/ccd

Dictionary (BIRD)[2], both obtained from the RCSB Protein Data Bank (PDB). As of February 2024, the CCD dataset contained 45,622 molecules, primarily small organic compounds commonly found in macromolecular crystallography, while the BIRD dataset contained 819 molecules, representing biologically relevant non-polymeric entities. These datasets were selected for their structural diversity and derivation from real protein–ligand systems. They include a number of challenging symmetric or pseudo-symmetric structures, which we analyze in detail through specific case studies in Section 6.

2.1 Data Preprocessing

Since the datasets were originally in different formats, we generated a new conformation for each entry, saved them in the SDF file format for further processing, and subsequently merged both datasets.

Initial conformer generation was primarily performed using the EmbedMolecule function of the RDKit toolkit [6], followed by structural optimization with the MMFF94 force field [7]. RDKit was chosen due to its efficient 3D embedding algorithm, improved handling of torsional strain, and its ability to generate high-quality conformers that are more physically realistic. In cases where RDKit's conformer generation failed, OpenBabel's conformer generation was used as a fallback due to its broader support for certain chemical structures and alternative embedding methods. Entries for which both tools failed to generate conformers were excluded from the dataset. Additionally, molecules containing fewer than five heavy atoms were removed to ensure structural relevance and meaningful molecular modeling.

After preprocessing, the final dataset comprised 45,706 molecules. An overview of the dataset is provided in Table 1.

Table 1: Overview of Molecule Sources.

Source	Molecules Retrieved	Molecules Retained	Conformer Generation Tool	
			RDKit	Openbabel
CCD	45622	44901	44630	271
BIRD	819	805	755	50
Total	46441	45706	45385	321

2.2 Conformer Generation

To generate realistic 3D conformations of molecules for downstream analysis (see Section 4), we employed **SMINA**[8], a fork of AutoDock Vina, using structure-based docking against a protein target.

The chosen target was **HIV-1 protease** from PDB entry **1EBY**, selected for the following reasons:

[2] https://www.wwpdb.org/data/bird

- **Symmetrical Dimeric Structure**: HIV-1 protease functions as a symmetrical homodimer, which mirrors the structural symmetry observed in many small molecules, making it a relevant docking environment.
- **Large Binding Pocket**: The active site is spacious and capable of accommodating a wide variety of ligand sizes, supporting the diversity of our dataset.

For the docking simulations, default parameters were used with one exception: the exhaustiveness setting, which determines the thoroughness of the search, was reduced from the default value of 8 to 4 to obtain results within a reasonable computation time.

For each ligand, up to nine docked conformations were generated and saved in a single SDF file. These conformations were subsequently used for downstream analyses, including symmetry evaluation and conformational clustering.

2.3 Final Data Format and Organization

To ensure compatibility with various RMSD calculation tools, including ***DockRMSD***, the dataset underwent the following formatting and organization steps:

- **Conversion to MOL2 Format:** All SDF files containing multiple conformations per molecule were converted to MOL2 format using the obabel tool from Openbabel toolkit, ensuring broad compatibility with RMSD tools.
- **Individual Conformation Files:** In addition to multi-conformer files, separate files for each conformation were generated in both SDF and MOL2 formats to facilitate structure-specific analyses.

The dataset is systematically organized to provide clear accessibility:

- **Parent Directories:** Molecules are categorized based on their source repository:
  - `CCD/[MOLECULE_ID]/`
  - `BIRD/[MOLECULE_ID]/`
- **Per-Molecule Subdirectories:** Each molecule is stored in a folder named after its unique identifier, which contains the following files:
  - `all_poses.sdf` – Multi-conformation file in SDF format.
  - `all_poses.mol2` – Multi-conformation file in MOL2 format.
  - `pose_X.sdf` – Individual conformation X in SDF format.
  - `pose_X.mol2` – Individual conformation X in MOL2 format.

This structured approach ensures efficient data retrieval, compatibility with docking validation tools, and seamless RMSD analysis across different molecular modeling workflows.

2.4 Statistical Analysis of Benchmark Molecules

To better understand the composition and structural diversity of our dataset, we performed a statistical analysis focusing on problem-related molecular properties like heavy atom count

distribution, distinct atom types count distribution, and also combinatorial properties like automorphisms count distribution. The results provide a comprehensive overview of the dataset's characteristics, aiding in molecular modeling and cheminformatics applications.

Heavy Atom Count Distribution: As noted earlier, atom count significantly impacts the computational complexity of molecular comparison tasks. However, hydrogen atoms are typically omitted in RMSD calculations, making heavy atom count a more relevant metric. In our dataset, heavy atom counts range from 5 to 244, reflecting a wide range of molecular sizes. The majority of molecules, however, contain fewer than 50 heavy atoms, indicating a concentration of compact, chemically meaningful structures (Fig .1).

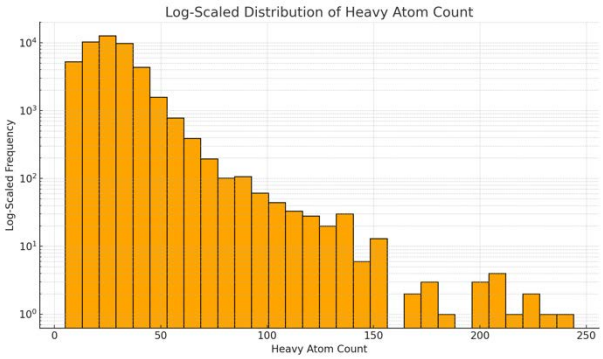

Fig. 1. Log-scaled distribution of heavy atom counts across the dataset.

Distinct Atom Types Count Distribution: While not as directly influential as total or heavy atom counts, the number of distinct atom types in a molecule can affect RMSD calculations by increasing the number of potential matching groups. In our dataset, this value typically ranges from 3 to 6, with a maximum of 8 (Fig. 2), reflecting a moderate yet meaningful degree of elemental diversity. This variation further supports the structural richness and chemical diversity of the dataset.

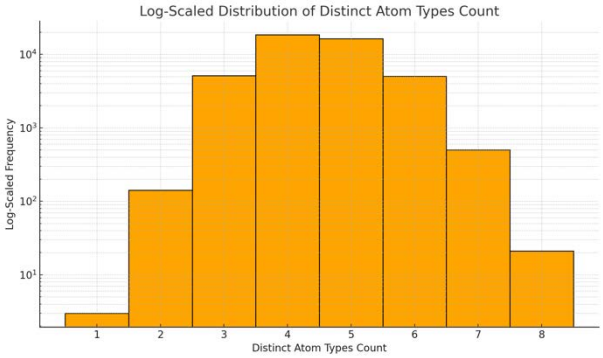

Fig. 2. Log-scaled distribution of distinct atom type counts across the dataset.

Automorphisms Count Distribution: We used the latest version (2.8.9) of the **dreadnaut** tool from nauty&Traces [9] toolset to quantify molecular symmetry. Graph representation files were generated for all molecular structures, which were then processed using **dreadnaut** to compute the number of automorphisms for each molecule. The resulting statistics are summarized in Fig. 3. Notably, a large portion of the dataset falls into the <2, 2–5, and 5–10 bins. Molecules with

moderate symmetry (10–100 automorphisms) form a secondary peak, while highly symmetric structures (over 1000 automorphisms) are rare.

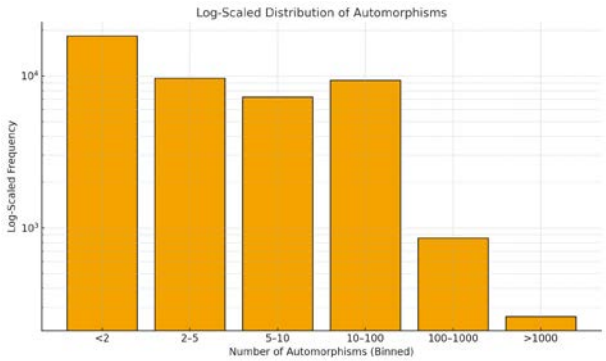

Fig. 3: Log-scaled distribution of the number of automorphisms across the dataset.

3. FlashRMSD Tool

3.1 Overview

The ***FlashRMSD*** tool is designed for efficient and robust symmetry-corrected RMSD calculations, supporting multiple molecular file formats including SDF, MOL, and MOL2, as well as files containing multiple conformations. It accommodates both standard and advanced use cases through a comprehensive set of configurable options.

The tool provides several key features:

- **Naïve calculation (-n flag):** Runs naïve search, by iterating over all permutations of possible matching atom groups. Can be used for results validation.
- **Hydrogen inclusion (-h flag):** Includes hydrogen atoms in RMSD calculations.
- **Bond order enforcement (-b flag):** Ensures strict bond order matching during atom mapping, preserving chemical integrity. This deterministic feature distinguishes ***FlashRMSD*** from other tools by enforcing chemically valid matches.
- **Verbose and assignment output (-v, -a flags):** Provides detailed runtime diagnostics and atom-to-atom assignment outputs for in-depth analysis.
- **Cross-RMSD calculation (-x flag):** Computes pairwise RMSD across all conformations within a single file, similar to the functionality of ***obrms***.
- **Multi-query input support:** Allows a reference conformation (first structure in a template file) to be compared against all conformations in a query file, enabling batch comparison workflows.

3.2 Algorithm

FlashRMSD utilizes a two-stage approach that combines descriptor-based atom featurization with an optimized backtracking algorithm to achieve symmetry-aware atom mapping.

Stage 1: Atom Descriptor Generation

Each atom is encoded with a descriptor array created via breadth-first traversal of the molecular graph starting from that atom as a root. For each traversed atom, its periodic table number and the distance from the root atom are encoded into a single integer (descriptor) using the formula:

$$DescriptorEncodedValue = 2^{10} \cdot distance + PeriodicTableNumber$$

The resulting descriptor arrays for each atom are sorted and eventually hashed into a single integer. Hashing is used to avoid costly array comparisons; thus, any consistent function can be used. The encoding formula ensures that after sorting a descriptor array, descriptors from the same BFS (Breadth-First Search) layer occupy adjacent positions. This has the same effect as if we kept an array of descriptor arrays per distance from the root atom. In this way, a sorted descriptor array effectively encodes all level neighborhood information, and so does its hash. As these values encode BFS layers' information, we'll refer to them as **Layer Data**.

Although this stage has a complexity of $O(C \times N^2)$ (where C is the number of conformations and N the number of atoms), it lays the groundwork for efficient atom mapping. This approach becomes particularly advantageous during cross-RMSD calculations. In a standard RMSD comparison between two conformers, only two featurizations and one RMSD calculation are required. However, in cross-RMSD mode, the process involves C featurizations followed by $C(C-1)/2$ RMSD computations. As the number of conformers increases, the computational load shifts from featurization to RMSD calculation, highlighting the importance of optimizing the latter.

Here's an example of how atom descriptors are generated for a single **O** atom of $\mathbf{SO_4}$ molecule (see Fig. 4).

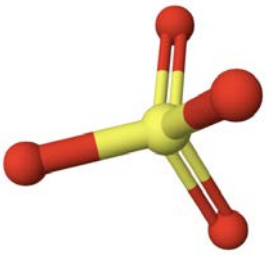

Fig. 4. $\mathbf{SO_4}$ molecule, with the sulfur atom shown in yellow and oxygen atoms in red.

During BFS traversal, we'll visit **O** atom at distance 0, **S** atom at distance 1, and 3 more **O** atoms at distance 2, thus **O** atoms descriptor will have the following value:

$$hash([0+8, 1024+16, 2048+8, 2048+8, 2048+8]) = 1428496640$$

Stage 2: Atom Mapping via Backtracking

The following algorithm is presented for mapping atoms between a pair of conformations, after the atom descriptor generation stage is completed for each:

- Candidate lists are generated for each atom in the first conformation based on descriptor matches.

- An optimized backtracking search is performed to determine the best atom-to-atom mapping, with the following optimization levels:
    - **Level 1:** Naive backtracking using all candidates.
    - **Level 2:** Trivial one-to-one matches are resolved and removed before backtracking to reduce complexity.
    - **Level 3 (default):** After excluding trivial matches, atoms are grouped into independent blocks using a Disjoint Set Union (DSU) based on descriptor matches or bonding. Each block is processed independently, and results are combined for the final mapping.

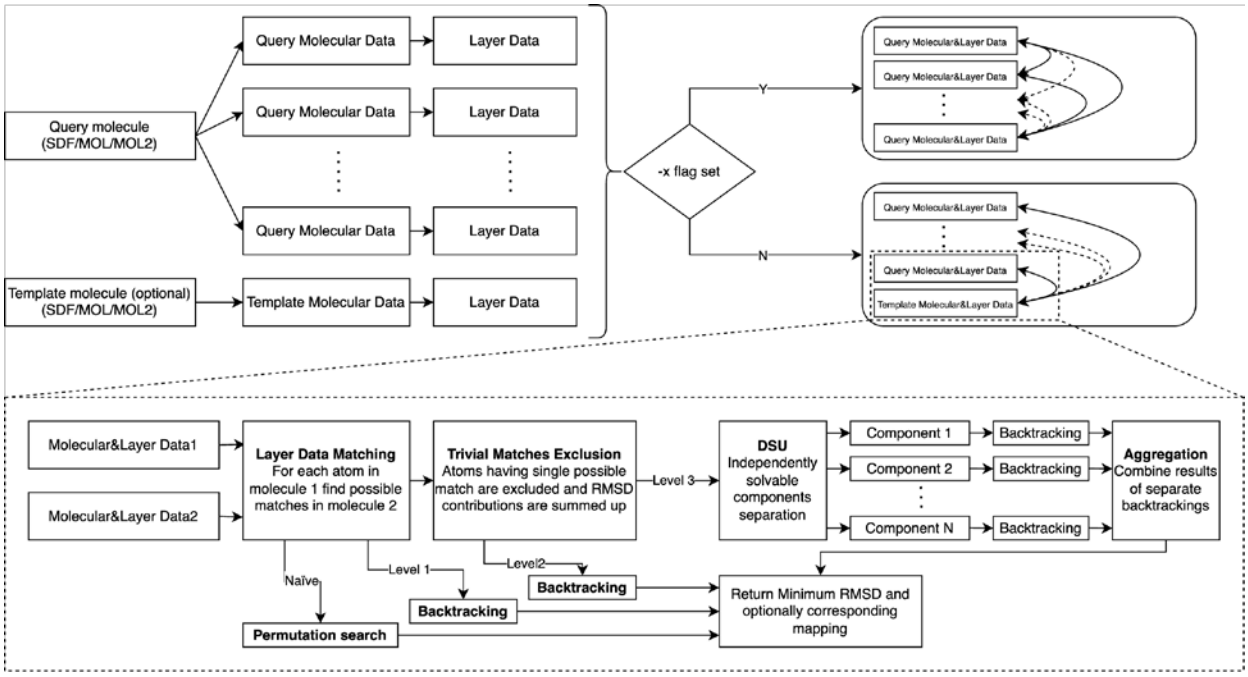

Fig. 5. Flowchart of the FlashRMSD Algorithm.

This flowchart illustrates the two main stages of the ***FlashRMSD*** algorithm (see Fig. 5). The first stage includes the atom featurization, where each atom's descriptor is generated through a breadth-first traversal and hashed to produce a unique fingerprint, and computation mode determination. The second stage depicts the pairwise mapping process: candidate list generation based on matching descriptors, followed by optimizations including trivial mapping exclusion and block decomposition, and finally, the backtracking procedure used to derive the optimal mapping.

4. Benchmark Setup

While RMSD is defined between two molecular conformations, in practical applications, especially within automated pipelines, it is usually computed across multiple conformations. For instance, in docking workflows, multiple binding poses are often generated and must be compared with each other to identify distinct clusters. This step typically precedes more expensive stages such as rescoring or molecular dynamics, making early-stage correctness and robustness crucial. Therefore, a more scalable interface for cross-RMSD calculations is often more important. Tools

that support efficient, reliable cross-RMSD interfaces better support real-world use cases such as clustering, redundancy filtering, and structural diversity analysis.

Additionally, when RMSD tools are used repeatedly or integrated into long-running workflows, even minor issues, such as memory leaks, crashes, or incorrect output, can propagate and cause significant downstream errors. Therefore, we argue that benchmarking tools on their cross-RMSD functionality is not only representative of real usage scenarios but also a more comprehensive test of tool robustness and interface design**.**

In our benchmarking, we consider two complementary setups:

1. **Cross-RMSD Native Benchmark** – Tools are tested on their ability to compute all-pair RMSD values across a set of poses through their native interface.
2. **All-to-All Pairs RMSD Benchmark** – Tools are also tested on computing the same RMSD matrix using repeated two-pose calls to simulate scenarios where no cross-RMSD interface is available.

Tools will be evaluated along three criteria:

- **Reliability**: Success rate across tasks, accounting for errors, crashes, indefinitely long runtime, or invalid outputs.
- **Correctness**: Agreement with reference calculations using naïve but accurate RMSD implementations.
- **Performance**: Execution time, measured only on cases where all tools succeed to ensure fair comparison.

This setup allows us to assess both the core computational correctness and the practical utility of RMSD tools in scalable scientific applications.

To ensure a fair and meaningful comparison across tools, we extended the functionality of ***DockRMSD*** in two key ways. First, we modified the tool to support cross-RMSD computation directly from a single multi-conformer input file. This significantly reduces the number of redundant pairwise calls and mitigates file I/O overhead, aligning ***DockRMSD***'s interface more closely with tools like ***obrms*** and ***FlashRMSD*** that natively support cross-RMSD calculations.

Second, we addressed limitations in ***DockRMSD***'s file parsing. The original implementation only supported a narrow subset of MOL2 files, rejecting valid inputs that deviated from expected formatting. We revised the file reading logic to accommodate a broader range of MOL2 variants by relaxing strict constraints and improving parser robustness. These changes eliminate unnecessary preprocessing steps and ensure compatibility with more diverse datasets, improving ***DockRMSD***'s utility in real-world workflows.

The resulting extended version, supporting both cross-RMSD input handling and enhanced MOL2 compatibility, is referred to as ***DockRMSDExt*** in our benchmarks. This ensures that performance and reliability comparisons across tools reflect differences in computational design, rather than constraints imposed by tool interfaces or input formatting.

4.1 Benchmarking Environment and Tools

To ensure fair and reproducible comparison across RMSD calculation tools, all benchmarks were conducted on a consistent hardware and software environment with the following specifications:

- **CPU**: AMD EPYC 9654 96-Core Processor
- **RAM**: 504 GB DDR4
- **Operating System**: Ubuntu 22.04 LTS (64-bit)
- **Storage**: NVMe SSD
- **Python Version**: 3.12.9 (used for automation, validation, and timing)

Each benchmarking task was run as a separate process to avoid system-level interference, and wall-clock times were measured using Python-based orchestration scripts. All tools were tested using their latest stable versions, compiled with default settings where applicable.

We evaluated the following tools:

- **obrms**
- **FlashRMSD (Level 3)**
- **FlashRMSDNaive**
    - FlashRMSD tool with naïve flag set, iterates over all possible mappings after layer data matching (**Figure 5**)
- **DockRMSD**
- **DockRMSDExt**

In this benchmark, we exclude ***spyRMSD*** due to its prohibitively slow performance and prior evidence of inefficiency [1, 2], focusing instead on faster tools for runtime evaluation.

5. Results

5.1 Cross-RMSD Native Benchmark

This benchmark focuses on evaluating each tool's capability to compute all-to-all RMSD values across multiple conformations of the same molecule using their native cross-RMSD interfaces, where available. This use case is central to workflows that require clustering or structural deduplication prior to downstream analysis or simulation.

As mentioned before, ***DockRMSD*** doesn't provide a native interface for such calculations, thus, we'll compare other tools against each other.

For this benchmark, each tool was provided with a single MOL2 file containing multiple conformations of the same molecule. The expected output was a complete pairwise RMSD matrix of size $N \times N$, where N is the number of conformations in the input. Only the upper triangular part (excluding the diagonal) was used for performance analysis, as RMSD matrices are symmetric.
To assess correctness, outputs were compared against results from ***FlashRMSDNaive***, which performs exhaustive symmetry correction without heuristics. Minor floating-point differences

were allowed within a predefined tolerance (0.001). Any discrepancies beyond this threshold were flagged and analyzed.

To prevent excessive runtimes from affecting the benchmark, a per-call timeout of 60 seconds was set. Any individual RMSD computation that exceeded this limit was recorded as a timeout failure. However, for naïve calculations, the timeout was set to 180 seconds.

Runtime was measured for successful runs only, using wall-clock time recorded externally via orchestration scripts. This benchmark isolates and evaluates tools specifically on their native ability to handle structured, multi-conformer input efficiently and correctly.

Out of 45,706 total samples, 45,543 were completed successfully across all tools. For the remaining 163 samples, only timeout-related failures were encountered—no runtime crashes or output corruption were observed. We also verified that all outputs from the tools were numerically identical for the successful cases.

Table 2. Runtime summary of symmetry-corrected RMSD calculation tools on cross-RMSD benchmark (45,543 samples)

Tool	**Mean (s)**	**Std (s)**	**Min (s)**	**Max (s)**
FlashRMSD	0.0137	0.0099	0.0041	0.4596
FlashRMSDNaive	0.0736	2.3091	0.0074	169.3351
DockRMSDExt	0.0510	0.8490	0.0043	55.0944
obrms	0.0571	0.7833	0.0206	47.0439

As shown in Table 2, ***FlashRMSD*** outperformed all other tools in terms of runtime, completing tasks approximately 4 times faster than its nearest competitor on average.

For the 163 samples where one or more tools failed, we analyzed the output of ***FlashRMSD*** on the same cases. Notably, ***FlashRMSD*** failed for only 7 samples, all of which also failed in other tools. For the remaining cases where only other tools failed, ***FlashRMSD*** completed successfully, and its output matched with the succeeding tools.

Table 3. FlashRMSD runtime on samples that failed in other tools.

Failed Tool	**Number of failures**	**FlashRMSD runtime report**		
		Mean (s)	Min (s)	Max (s)
FlashRMSDNaive	43	1.8886	0.0064	49.7958
DockRMSDExt	118	0.4810	0.0056	49.7958
obrms	36	2.3881	0.0063	49.7958

As seen in Table 3, ***FlashRMSD*** handled most of these challenging samples well, maintaining reasonable runtimes. However, a single outlier pushed its maximum runtime to 49.8 seconds, which was close to the timeout threshold. This suggests the tool is generally robust, with rare edge cases that may require monitoring.

5.2 All-to-All Pairs RMSD Benchmark

This benchmark evaluates the behavior and performance of RMSD calculation tools when used in pairwise mode, computing RMSD values between all unique pairs of conformations. Unlike the native cross-RMSD benchmark, this approach requires invoking the tool separately for each pose pair, simulating the fallback strategy often required by tools that lack native cross-RMSD support.

For a molecule with N conformers, this results in $N(N-1)/2$ individual RMSD computations. All tools were orchestrated via automated scripts to execute these comparisons sequentially, and per-call runtimes were collected. For this benchmark, a timeout of 5 seconds per call was set; any computation exceeding this limit was considered a timeout failure.

The objectives of this benchmark are threefold:

- To enable a direct comparison with the original ***DockRMSD***, which does not support native cross-RMSD and must operate in this mode by design.
- To evaluate robustness and failure rates across specific pairwise comparisons, especially in challenging edge cases.
- To identify and showcase individual pose pairs for which certain tools fail, providing insight into tool stability and error patterns.

All available tools, including those with native cross-RMSD support, were evaluated in this benchmark to ensure a uniform baseline for comparison. As in the cross-RMSD benchmark, ***FlashRMSDNaive*** was used as the reference for correctness verification.

Out of 45,706 total samples, 42,406 were successfully processed by all tools, including the original implementation of ***DockRMSD***. However, when excluding ***DockRMSD***, the number of successful samples increases to 45,558. This discrepancy is due to the file parsing limitations of the original ***DockRMSD*** implementation, as discussed earlier.

On all samples where any two tools produced results, their outputs were in agreement in terms of correctness. To evaluate whether our modified version—***DockRMSDExt***—can reliably replace ***DockRMSD*** in broader benchmarks, we compared the two implementations on the 42,406 samples that both completed successfully.

Table 4. Runtime comparison of DockRMSD and DockRMSDExt on all-to-all pairs benchmark (42,406 samples)

Tool	**Mean runtime over all calls (s)**	**Mean of per-sample averages (s)**	**Std over all calls (s)**	**Std of per-sample averages (s)**
DockRMSD	0.00573	0.00562	0.0439	0.0367
DockRMSDExt	0.00571	0.00560	0.0442	0.0368

As shown in Table 4, the runtime performance of ***DockRMSD*** and ***DockRMSDExt*** is nearly identical. In fact, the revised version is marginally faster on average. This indicates that the improvements to file parsing in ***DockRMSDExt*** do not introduce any runtime penalty, validating its use in place of the original implementation.

Moreover, the original ***DockRMSD*** failed on approximately 7% of the total dataset due to strict file parsing issues—failures that are fully resolved in ***DockRMSDExt***. Therefore, we will use ***DockRMSDExt*** in all further benchmarks as a reliable and representative version of ***DockRMSD***. We conducted the same benchmark as in the previous section, with one key difference: we evaluated both per-call runtimes across all pose pairs and per-sample average runtimes separately to capture different aspects of tool performance.

Table 5. Runtime summary of symmetry-corrected RMSD calculation tools on all-to-all pairs benchmark (1,458,326 pairs)

Tool	**Mean (s)**	**Std (s)**	**Min (s)**	**Max (s)**
FlashRMSD	0.00435	0.00176	0.00150	0.06144
FlashRMSDNaive	0.00557	0.06241	0.00164	4.93368
DockRMSDExt	0.00607	0.04947	0.00170	4.94237
obrms	0.02072	0.02414	0.01431	2.47477

As shown in Table 5, ***FlashRMSD*** consistently outperforms other tools in terms of runtime on this benchmark. Notably, the minimum runtime for ***obrms*** is significantly higher than the other tools, reflecting the inherent overhead associated with being part of a larger, more complex codebase.

A more comprehensive view of runtime distributions across all tools can be seen in Fig. 6, which presents the box-and-whisker plot of per-call runtimes for all 1,458,326 comparisons.

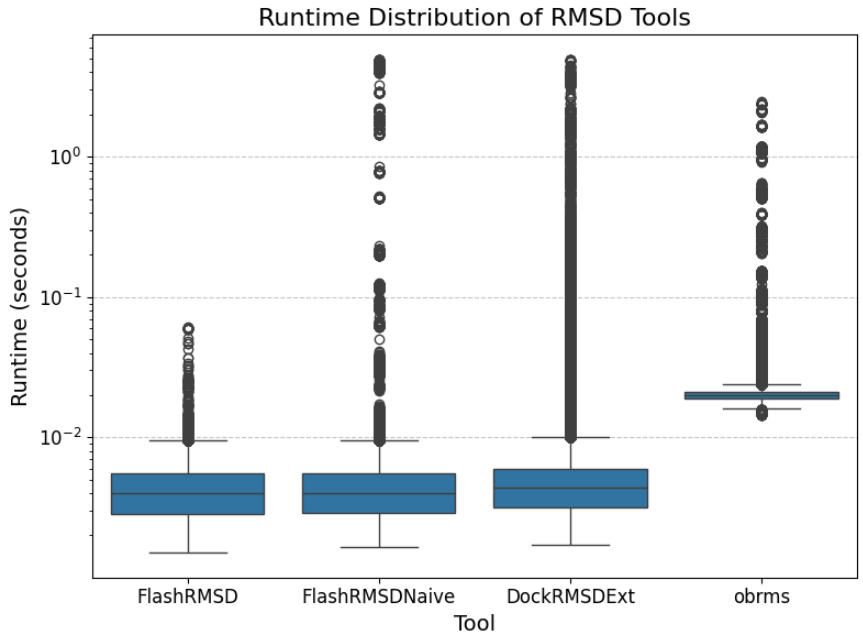

Fig. 6. Box and whiskers plot of runtimes of symmetry-corrected RMSD calculation tools (1,458,326 pairs)

Table 6. Per-sample average runtime summary of symmetry-corrected RMSD calculation tools on all-to-all pairs benchmark (45,558 samples)

Tool	Mean (s)	Std (s)	Min (s)	Max (s)
FlashRMSD	0.00435	0.00174	0.00162	0.04436
FlashRMSDNaive	0.00544	0.05885	0.00188	4.72553
DockRMSDExt	0.00593	0.04068	0.00191	3.50721
obrms	0.02063	0.02335	0.01563	2.38896

The results in Table 6 further support the conclusion that ***FlashRMSD*** outperforms other tools in terms of per-sample average runtime. This demonstrates that the complex atom featurization used in our algorithm, originally introduced to optimize cross-RMSD calculations, does not introduce any runtime overhead when applied to pairwise RMSD computations. On the contrary, ***FlashRMSD*** remains the most efficient across both benchmark modes.

Finally, there were 148 samples where one or more tools failed during the all-to-all pairwise benchmark. ***FlashRMSD*** failed on the fewest samples — 5 in total and consistent with previous results, all other tools also failed on those 5 samples.

Table 7. FlashRMSD runtime on pairs that failed in other tools.

Failed Tool	Number of failed samples	Number of failed pairs	FlashRMSD runtime report		
			Mean (s)	Min (s)	Max (s)
FlashRMSDNaive	40	1370	0.1530	0.0017	3.7507
DockRMSDExt	106	2932	0.0656	0.0018	3.7507
obrms	9	324	0.1101	0.0021	2.1288

As shown in Table 7, ***FlashRMSD*** handled these challenging samples successfully, maintaining reasonable runtime performance even in cases where other tools failed.

6. Case Studies

In this section, we'll dive into benchmark results focusing on interesting molecules discussed in [1, 3], and also two new challenging examples identified during our current benchmarks.

CCD/PE3, CCD/33O

The molecules PE3 and 33O, previously discussed in [1], are known to consistently cause failures in the original ***DockRMSD*** implementation. Both structures consist of chains of alternating carbon and oxygen atoms, creating symmetric topologies that introduce multiple valid atom mappings during alignment.

These systems are particularly interesting because they expose limitations in tools that rely heavily on strict atom ordering or lack robust symmetry handling. In both cases, all tested tools, except for ***DockRMSD***, successfully completed the RMSD calculation within the time limit. ***DockRMSD*** consistently exceeded the 5-second timeout, failing to return results.

Table 8. Comparison of RMSD calculation tools on DockRMSD breaking samples. Per-pair average runtimes are presented in seconds.

Tool	**PE3 (per-pair average) (s)**	**33O (per-pair average) (s)**
FlashRMSD	0.00282	0.00615
FlashRMSDNaive	0.00313	0.00563
obrms	0.02026	0.02033

As shown in Table 8, all successful tools returned results in a fraction of a second. The ***FlashRMSD*** and ***FlashRMSDNaive*** runtimes are nearly identical, but notably, ***FlashRMSDNaive*** performs slightly faster than the optimized implementation in the case of **33O**. This rare case emphasizes that while general optimizations are effective, atom featurization and initial pruning strategies are critical for performance consistency. Poorly suited heuristics or inadequate pruning, especially in highly symmetric cases, can lead to exhaustive search behavior even in otherwise optimized tools.

CCD/60C

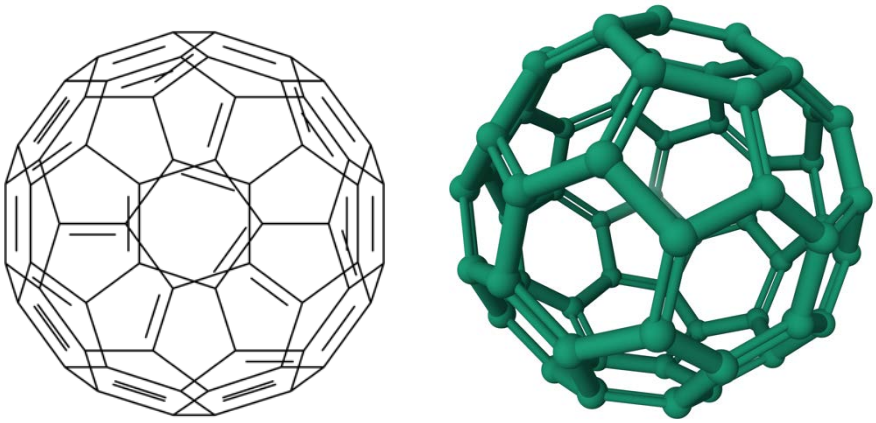

Fig. 7. 60C (buckminsterfullerene) molecules 2D (left) and 3D (right) structures.

The molecule **60C** (Fig. 7), previously analyzed in [3] for comparison between ***DockRMSD*** and ***obrms***, serves as a valuable case for evaluating tool performance under extreme symmetry. Here, we extend the analysis by including benchmark results from the ***FlashRMSD*** and ***FlashRMSDNaive*** tools.

Structurally, **60C** features **12** pentagonal and **20** hexagonal faces arranged in a fullerene-like topology. A critical detail is that every edge of a pentagonal face is shared with a hexagonal face. This edge-sharing relationship creates a unique fingerprint for certain bonds; specifically, edges that bridge a pentagon and a hexagon are uniquely identifiable, as they cannot be matched to bonds lying solely between two hexagons.

As a result, when attempting to match two conformers of **60C**, any mapping that aligns a bond connecting a pentagon and a hexagon in the template must align with the corresponding bond in the reference. This significantly constrains the mapping space and leads to $2 \times 60 = 120$ possible

mappings — a manageable number, in contrast to estimates in [3]. However, the challenge lies in efficiently searching and pruning this space.

Despite this manageable mapping space, the ***FlashRMSDNaive*** tool failed to compute RMSD for any pose pairs, highlighting the limitations of exhaustive, non-pruned search methods in symmetric systems. The other tools, however, successfully completed the calculations and yielded the following average runtimes:

- ***FlashRMSD***: 5.8 ms
- ***DockRMSD***: 12.3 ms
- ***obrms***: 37.6 ms

These results demonstrate that ***FlashRMSD*** outperforms both ***DockRMSD*** and ***obrms***, achieving approximately **2** and **6.5** times better runtimes, respectively. The case of **60C** underscores the importance of efficient pruning and symmetry-aware mapping strategies, even in search spaces that are theoretically tractable. Without such optimizations, tools can still struggle or fail under the computational weight of redundant mappings.

BIRD/PRDCC_900031

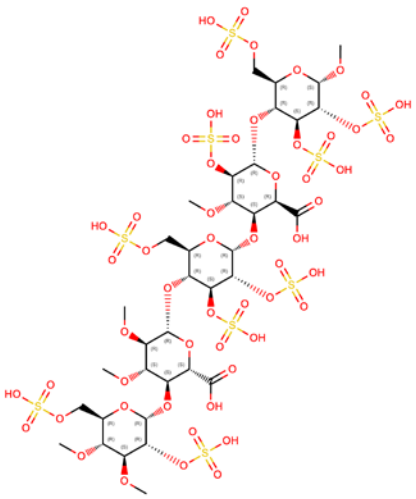

Fig. 8. PRDCC_900031(heparin pentasaccharide) molecules 2D structure.

The molecule **PRDCC_900031** (Fig. 8) serves as a prime example where all key design features of the ***FlashRMSD*** tool contribute directly to performance. At a glance, the molecule appears to have a symmetric scaffold due to its ring-chain architecture and repetitive **SO** $_4$ (sulfate) or **COOH** substituents. However, a closer inspection reveals that the core scaffold is not symmetric: the rings contain alternating carbon and oxygen atoms in a way that breaks symmetry.

Thanks to its advanced atom featurization, ***FlashRMSD*** is able to quickly detect this and identify a trivial atom mapping, effectively ruling out unnecessary branches during backtracking. This dramatically improves performance.

The real complexity arises from the nine **SO** $_4$ and two **COOH** groups attached to the leaf atoms of the backbone. Each **SO** $_4$ group can be matched in 3! (6) different ways, and each **COOH** group can be matched in 2! (2) ways, leading to a theoretical explosion of $2^2 6^9 = 40{,}310{,}784$

possible mappings across the entire molecule. While other tools treat this as a flat, unstructured mapping problem, ***FlashRMSD***'s level-3 optimization decomposes the problem: each symmetrical group mentioned above is treated as an independent subtree, allowing mappings to be computed separately and then combined. This reduces the mapping search space from $2^2 6^9$ to just $2 \times 2 + 9 \times 6 = 58$ evaluations, a drastic and principled reduction.

All other tools failed to compute cross-RMSD in a reasonable time. ***FlashRMSDNaive***, despite correctly accounting for symmetry, was forced to iterate through the full $2^2 6^9$ mappings, completing in 46.9 seconds. In contrast, ***FlashRMSD*** completed the same calculation in just 8.9 milliseconds, clearly demonstrating the power of intelligent symmetry decomposition.

We also evaluated all-to-all pairwise RMSD performance. ***DockRMSD*** succeeded on only 4 out of 36 pairs, while ***obrms*** failed on all. This case illustrates that clever partitioning of symmetric substructures can make the difference between exponential runtime and milliseconds.

CCD/7AZ, CCD/FWQ

These molecules are special because, independently, they managed to fit into a 5-second window, but in the cross-RMSD benchmark, the total runtime was bigger than 60 seconds. They both have a similar structure and represent a special case of symmetries – a big macrocycle with trailing similar components from macrocycle nodes. These kinds of samples are the subject of investigation as how they can be effectively analyzed.

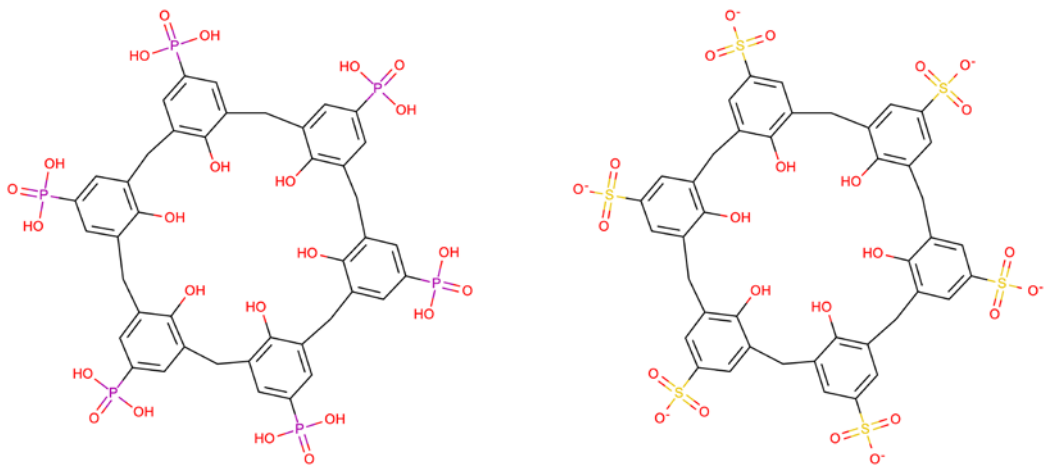

Fig. 9. 7AZ (left) and FWQ (right) molecules 2D structures.

Fig. 9 shows that both molecules share a common motif: a large macrocyclic core with branching, symmetry-repeating fragments extending from multiple macrocycle nodes. These fragments exhibit local similarity, but are distributed across the molecular structure in ways that significantly increase the number of potential atom mappings. These examples suggest a new class of test cases that require new approaches in future RMSD tools.

7. Conclusions

This work presents ***FlashRMSD***, a symmetry-corrected RMSD calculation tool designed for accuracy, efficiency, and robustness in both standalone and large-scale automated workflows. Alongside ***FlashRMSD***, we introduce a comprehensive benchmark dataset comprising thousands of molecular pose comparisons, specifically structured to evaluate tool performance under realistic and challenging scenarios.

Through systematic benchmarks, including native cross-RMSD calculations and all-to-all pairwise comparisons, we demonstrate that ***FlashRMSD*** consistently outperforms existing tools in terms of runtime, reliability, and correctness. It exhibits superior scalability, maintaining low variance across diverse molecular structures, and handles failure-prone or highly symmetric cases with resilience. Importantly, the optimizations introduced for cross-RMSD efficiency do not introduce overhead in simpler pairwise use cases.

Our benchmark suite also highlights structural motifs that pose challenges to current RMSD tools, such as highly symmetric systems, macrocyclic architectures, and molecules with repetitive substructures or symmetric side chains. These special cases, analyzed in detail, provide insight into where existing tools struggle and where future development should focus.

We make both ***FlashRMSD*** and the full benchmark dataset publicly available to facilitate reproducible evaluation and guide future development of RMSD tools. We hope this contribution will support more reliable and scalable structural comparison workflows in molecular modeling, docking, and related fields.

Appendix

Data and Code Availability

The benchmark dataset developed for this study is publicly available via Zenodo at https://doi.org/10.5281/zenodo.15097621. It includes over 45,000 small molecules from the CCD and BIRD repositories, complete with multi-conformer and per-pose files, as well as a results.csv file containing ground truth cross-RMSD values.

The source code for the ***FlashRMSD*** tool, along with the modified tool ***DockRMSDExt***, is available on GitHub at https://github.com/altunyanv/FlashRMSD. Both the dataset and the code are released under open-source licenses to facilitate reproducibility and further development in symmetry-corrected RMSD calculations.

References

[1] V. Altunyan, “Comparative analysis of symmetry corrected rmsd calculation tools in molecular docking”, *Vestnik RAU*, vol. 1, pp. 25-36, 2024.

[2] R. Meli and P. C. Biggin, “spyrmsd: symmetry-corrected RMSD calculations in Python”, *Journal of Cheminformatics*, vol. 12, no. 49, 2020. https://doi.org/10.1186/s13321-020-00455-2

[3] E.W. Bell and Y. Zhang, “DockRMSD: an open-source tool for atom mapping and RMSD calculation of symmetric molecules through graph isomorphism”, *Journal of Cheminformatics*, vol. 11, no. 40, 2019. https://doi.org/10.1186/s13321-019-0362-7

[4] N. M. O'Boyle, M. Banck, C.A. James, C. Morley, T. Vandermeersch and C. R. Hutchison, “Open Babel: An open chemical toolbox.”, *Journal of Cheminformatics*, vol. 3, no. 33, 2011. https://doi.org/10.1186/1758-2946-3-33

[5] J. D. Westbrook, C. Shao, Z. Feng, M. Zhuravleva, S. Velankar and J. Young, “The chemical component dictionary: complete descriptions of constituent molecules in experimentally determined 3D macromolecules in the Protein Data Bank”, *Bioinformatics*, vol. 31, no. 8, pp. 1274–1278, 2015.

[6] RDKit: Open-source cheminformatics. [Online]. Available: https://www.rdkit.org

[7] P. Tosco, N. Stiefl, G. Landrum, “Bringing the MMFF force field to the RDKit: implementation and validation”, *Journal of Cheminformatics*, vol. 6, no. 37, 2014. https://doi.org/10.1186/s13321-014-0037-3

[8] D. R. Koes, M.P. Baumgartner and C.J. Camacho, “Empirical scoring with smina from the CSAR 2011 benchmarking exercise”, *Journal of Chemical Information and Modelling*, vol. 53, pp. 1893-1904, 2013.

[9] B.D. McKay and A. Piperno, “Practical graph isomorphism, II”, *Journal of Symbolic Computation*, vol. 60, pp. 94-112, 2014.

FlashRMSD. Արդյունավետ մոտեցում սիմետրիայով ճշգրտված RMSD հաշվարկի համար և համապարփակ բենչմարք վերլուծություն

Վահագն Ն. Ալթունյան

Երևանի պետական համալսարան, Երևան, Հայաստան
e-mail: altunyanv@gmail.com

Ամփոփում

Արմատ միջին քառակուսային շեղումը (RMSD) առանցքային չափում է մոլեկուլային կառուցվածքների նմանությունը գնահատելու համար։ Սակայն սիմետրիկ մոլեկուլների դեպքում առաջանում են կոմբինատոր բարդություններ, որոնք խանգարում են հաշվարկի գործընթացին։ Թեպետ հասանելի են մի շարք գործիքներ, որոնք RMSD-ի հաշվարկում հաշվի են առնում սիմետրիաները, բոլորն էլ ունեն իրենց սահմանափակումները՝ կապված արագության, ճշգրտության կամ կիրառելիության

հետ: Այս հոդվածում մենք ներկայացնում ենք FlashRMSD գործիքը՝ նոր, արագ և արդյունավետ մոտեցում սիմետրիայով ճշգրտված RMSD հաշվարկի համար: Բացի այդ, մենք ներկայացնում ենք մոլեկուլային կառուցվածքների բազա RMSD գործիքների գնահատման և գործող մեթոդների համեմատական վերլուծության համար:

Բանալի բառեր՝ սիմետրիայով ճշգրտված RMSD, FlashRMSD, մոլեկուլային դոկինգ:

FlashRMSD: Эффективный подход к вычислению RMSD с учётом симметрии и расширенный бенчмарковый анализ

Ваагн Н. Алтунян

Ереванский государственный университет, Ереван, Армения
e-mail: altunyanv@gmail.com

Аннотация

RMSD является важным показателем для оценки сходства молекулярных структур. Однако при работе с сильно симметричными молекулами возникают комбинаторные сложности, которые затрудняют процесс вычислений. Хотя существует ряд инструментов с открытым исходным кодом для вычисления RMSD с учетом симметрии, каждый из них имеет ограничения по скорости, точности или удобству использования. В данной статье мы представляем FlashRMSD — новый, быстрый и эффективный метод вычисления RMSD с учетом симметрии. Также мы представляем обширный набор молекулярных структур для оценки инструментов вычисления RMSD и проводим сравнительный анализ существующих методов с нашим решением.

Ключевые слова: RMSD с коррекцией симметрии, FlashRMSD, молекулярный докинг.