Durandal: Fast Exact Clustering of Protein Decoys
Francois Berenger,[a] Rojan Shrestha,[a] Yong Zhou,[a] David Simoncini,[a] and Kam Y. J. Zhang*[a]

Introduction

In the field of in silico protein structure prediction, folding simulations often create a huge number of candidate structures. Exact clustering is commonly used in combination with energy filtering as a way to identify the best models.

The exact clustering algorithm follows two repeating steps. First, the cluster containing the structure with the maximum number of neighbors within a given cutoff value is found. Second, this cluster is removed from the set remaining to be clustered. Subsequent clusters are found by iterating these steps until the remaining set is empty. Exact clustering is used by several successful protein folding engines like Rosetta[1] and I-TASSER.[2] Exact clustering is used to shrink the population of generated models and can smooth imperfections in the free energy functions used to evaluate the quality of models.[3]

Exact clustering after energy filtering identified conformations better than using energy only.[3] Exact clustering was also used to determine if a protein folding simulation needs more sampling of the conformational space.[4] Exact clustering allowed to identify the best decoys from protein folding runs with the SPICKER software.[5] Fast but approximate clustering of decoys was investigated in the SCUD software by using a cheap to compute upper bound of root-mean-square distance (RMSD) after optimal superposition.[6] Hierarchical clustering was used in the hierarchical clustering of protein models (HCPM) software (http://biocomp.chem.uw.edu.pl/HCPM/) to partition the sampled conformational space[7,8] and to identify structures with a native-like topology.[9] Handling large data sets quickly was tackled in Calibur (http://sourceforge.net/projects/calibur/). Calibur’s algorithm is an assembly of three strategies: outlier decoys filtering, use of cheap to compute lower and upper bounds of RMSD and exploiting the metric property of RMSD.[11]

To the best of our knowledge, Durandal is the fastest software implementing exact clustering. Durandal uses an accelerated distance matrix initialization algorithm.[12] Filling this matrix is one of the most time-consuming part of many clustering algorithms. Our software was named in reference to a mighty sword from medieval French legends. The sword was owned by Count Roland[13] and is said to have cut the Roland’s breach in the Pyrenean mountains.

In this note, we describe the incorporation of an improved RMSD calculation method. We also describe the architecture and features provided by the software. We hope this note will enable more users to get familiar with this tool and help potential contributors in getting accommodated with its source code.

Method and Software Features

Initializing the distance matrix in an efficient manner is the key part of our approach. RMSD being a metric,[11] allows us to prune out computations by taking advantage of the triangular inequality by measuring distances from all decoys to only a few reference ones. In the case of exact clustering, knowing only a

[a] F. Berenger, R. Shrestha, Y. Zhou, D. Simoncini, K. Y. J. Zhang
Zhang Initiative Research Unit, Advanced Science Institute, RIKEN,
2-1 Hirosawa, Wako, Saitama 351-0198, Japan. Fax: (048) 467 8790
E-mail: kamzhang@riken.jp
† Present address: Software School of Dalian University of Technology, Economy and Technology Development Area, Dalian 116620, People’s Republic of China.
Contract/grant sponsor: RIKEN (“Initiative Research Unit” program), Japan
range the distance will fall into is enough in many cases. This range can be approximated and narrowed down using only additions and subtractions, which is faster than computing a RMSD.

The naive way to initialize the distance matrix needed for exact clustering is to compute RMSD for each distinct decoy pair. On \( n \) decoys, it would require the computation of \( n(n-1)/2 \) RMSDs.

Our method is lazy: it favors distance ranges over exact measures. We borrowed the lower and upper bounds of \( C_\alpha \text{RMSD} \) from Calibur\(^{[10]} \) and SCUD\(^{[6]} \) respectively. We refer to “insertion” of information into the distance matrix as the action of measuring all decoys against a new, randomly chosen reference decoy. We refer to “propagation” as the use of geometric constraints to propagate information from an insertion step to the whole distance matrix. Once all the distance matrix ranges are sharp enough, the algorithm can proceed to the cluster enumeration step.

The insertion and propagation steps are crucial parts of our algorithm. If \( A, B, \) and \( C \) are three decoys (assuming \( A \) was randomly chosen to be the new reference decoy). \( AB \) and \( AC \) are exactly measured (insertion step). We subsequently sort these distances in increasing order (let’s assume \( AB \leq AC \)). Finally, we can approximate \( BC \) as \( AC - AB \leq BC \leq AC + AB \) (propagation step).

To guarantee the smart initialization is faster than the naive method, we monitor at run-time the initialization speed and fall back to a “lazy” version of the naive distance matrix initialization when it is detected to be faster. This “lazy” completion of the distance matrix initialization sequentially tries lower bound, then upper bound and finally resolves to RMSD in case none of the previous trials succeeded to sharpen enough the range under consideration. This algorithm is given as a flowchart in Figure 1. A formal description can be found in Berenger et al.\(^{[12]} \)

Basic functionalities of the software can be accessed and parameterized via short options on the command line. More advanced features are triggered by long options.

Durandal can perform:
- Accelerated exact clustering, referenced as “smart” in the program.
- Standard exact clustering, referenced as “brute force” in the program.
- Semi-automatic clustering distance threshold finding. This can be useful in order to adapt the clustering threshold to the spread of the decoy set.
- User-defined clustering distance threshold.
- Saving clusters to a file.
- Listing clusters of size equal to the biggest one, referenced as “pole position” clusters in the program.
- Computing RMSD of each protein data bank (PDB) file to the first one in the input file list. This can be useful to rank some \textit{in silico} generated models against a known native structure for example.
- Controlling the number of clusters being displayed (three by default, all on request or any positive number). Limiting the number of clusters being displayed is wise as many clusters consisting of a single structure may exist depending on the clustering threshold that was used and the decoy set being analyzed.
- Providing help to the user.

In a near future, more clustering algorithms may be added to the tool, depending on our ability to accelerate them without approximation.

**Software Architecture**

The source code is organized around logical grouping of features. Major objects are created by composition and there is no usage of object inheritance. For performance reasons, some highly accessed class attributes have a public visibility and do not provide accessor member functions.

Here is a description of the most important classes in the software. The dependencies of these classes are shown in Figure 2.
Durandal: Fast Exact Clustering of Protein Decoys

Figure 2. Simplified dependency graph of Durandal. Arrows indicate use relation between components.

- **durandal.cc**: contains the main function and is the entry point to the program. Source code in this file is responsible for user interaction, options setup, starting the computation and printing out clusters at the end.
- **DistMatrix.cc**: stores the distance matrix. Contains code to perform fast matrix initialization using propagation of geometric constraints as well as the brute force method. Monitoring of entropy decrease speed is performed during the propagation step. It can trigger a switch to the final initialization strategy that is almost like brute force but takes advantage of RMSD bounds. Computation of plain RMSD, its lower and upper bounds and the caching of previous measures is also performed in this file. Semi-automatic threshold finding and discovery of clusters are also conducted here.
- **DistRange.cc**: contains distance range class. It encapsulates the RMSD range of each decoy pair. This is the class of the elements stored in the distance matrix. The correctness of created ranges is enforced at runtime. Creating a range with a lower bound greater than its upper bound would result in a fatal error printed out and the program stopped.
- **Singleton.cc**: provides storage of option values and guarantees their uniqueness and accessibility throughout the program. Utility functions are also declared in this file for convenience.
- **Stru.cc**: holds the result of parsing a protein structure file stored in the PDB format. This is one of the files that we borrowed from the Calibur software.\[10\]

Performance

We have sought to further enhance the performance of Durandal by examining the bottleneck in the computation. We focused our effort on improving the RMSD calculation because even when using a smart method to initialize the distance matrix, RMSD computations can still take up to 60% of the whole processing time. An increase in RMSD computation speed is crucial to many clustering methods.

Recently, a new method of calculating the RMSD between two protein structures using the quaternion-based characteristic polynomial (QCP)\[14\] method has been reported. It is a fast, numerically stable and accurate algorithm. It computes the minimum RMSD between two vector sets without having to figure out the rotation matrix to obtain the optimal superposition between them.

We have incorporated the QCP method for RMSD calculations into Durandal. We ran it intensively and compared its runtime in a benchmark with the previous RMSD routine in Durandal (an implementation of the Kabsch algorithm\[16\]). Results are shown in Figure 3. In this test, the newly incorporated QCP routine is 28% faster than the previous RMSD implementation in Durandal.

Figure 3. Performance comparison between implementations of the Kabsch and QCP algorithms for RMSD calculation in Durandal. Elapse times were averaged over three runs. The computation was performed inside a loop of one million iterations to measure a significant amount of time.

This amount of acceleration was expected: a detailed analysis of the QCP method, going down to the average number of floating point operations (FLOPs) required can be found in Theobald. In a complete clustering run on three decoy sets, this acceleration of RMSD computations translates into a 13% overall performance increase on average (Fig. 4).

Figure 4. Performance comparison of clustering using the Kabsch and QCP algorithms in Durandal. Decoys are from the semfold\[17\] decoy set. Clustering threshold was fixed to 3.0 Å in all cases. Elapse times were averaged over three runs. The % on top of each QCP bar is the acceleration gain.

We previously performed a comparison in speed between Calibur, SCUD,\[6\] and Durandal. Calibur is the fastest software we know of capable of performing exact clustering while SCUD is an approximate method for the same task. Execution times were measured on some I-TASSER decoys while varying the set size and results were averaged over five runs. This comparison has been updated and now includes results for the newly QCP-enabled version of Durandal (Fig. 5). In this test, QCP enabled
Figure 5. Speed comparison between Calibur, SCUD (an approximate method), Durandal and Durandal-QCP on 1shfA decoys at 1.5 Å cutoff. For clarity, only one slice in the 3D landscape created by varying the cutoff from 0.5 to 8 Å by 0.5 Å increment is shown. When considering the full cutoff range, Durandal is faster than Calibur in 98% of all cases. SCUD and Durandal-QCP results were computed only for this slice.

Durandal is 27% faster than the previous version. Because they take advantage of how decoys are arranged in space, both Calibur and Durandal have variable speed depending on the decoy set and the clustering threshold being used.

Conclusions

Durandal has successfully addressed the clustering speed problem, without falling into the easier task of giving a new but approximated version of the algorithm. The case where the pairwise distance matrix does not fit into memory is still challenging. The matrix size is a major barrier against using the exact clustering algorithm on very large data sets.

It is an interesting problem to adapt Durandal’s approach to other clustering algorithms without approximating them. This software note should ease the first step for researchers interested in extending the method we proposed.

Acknowledgments

We thank RIKEN, Japan, for an allocation of computing resources on the RIKEN Integrated Cluster of Clusters (RICC) system.

[1] R. Das, D. Baker, Ann Rev Biochem 2008, 77, 363.
[2] S. Wu, J. Skolnick, Y. Zhang, BMC Biol 2007, 3, 17.
[3] D. Shortle, K. T. Simons, D. Baker, Proc Nat Acad Sci USA 1998, 95, 11158.
[4] M. R. Betancourt, J. Skolnick, J Comput Chem 2001, 22, 339.
[5] Y. Zhang, J. Skolnick, J Comput Chem 2004, 25, 865.
[6] H. Li, Y. Zhou, J Comput Chem 2005, 26, 1189.
[7] D. Gront, A. Kolinski, Bioinformatics 2005, 21, 3179.
[8] D. Gront, U. H. E. Hansmann, A. Kolinski, Int J Quantum Chem 2005, 105, 826.
[9] S. Kmiecik, A. Kolinski, Biophys J 2008, 94, 726.
[10] S. Li, Y. Ng, BMC Bioinformatics 2010, 11, 25.
[11] B. Steipe, Acta Crystallogr Sect A 2002, 58, 506.
[12] F. Berenger, Y. Zhou, R. Shrestha, K. Y. J. Zhang, Bioinformatics 2011, 27, 939.
[13] L. Gautier, La chanson de Roland: texte critique, accompagné d’une traduction nouvelle et précédé d’une introduction historique. Partie 2.; A. Mame: Tours, 1872.
[14] D. L. Theobald, Acta Crystallogr Sect A 2005, 61, 478.
[15] P. Liu, D. K. Agrafiotis, D. L. Theobald, J Comput Chem 2010, 31, 1561.
[16] W. Kabsch, Acta Crystallogr Sect A 1978, 34, 827.
[17] R. Samudrala, M. Levitt, BMC Struct Biol 2002, 2, 3.

Received: 20 June 2011
Revised: 16 September 2011
Accepted: 11 October 2011
Published online on 26 November 2011