11-16-2017

Efficient pebble game algorithms engineered for protein rigidity applications

Mojtaba Nouri Bygi
Smith College

Ileana Streinu
Smith College, istreinu@smith.edu

Efficient pebble game algorithms engineered for protein rigidity applications

Mojtaba Nouri Bygi
Department of Computer Science
Smith College
Northampton, Massachusetts, USA
mnouribygi@smith.edu

Ileana Streinu
Department of Computer Science
Smith College
Northampton, Massachusetts, USA
istreinu@smith.edu

***Abstract*—Pebble game rigidity analysis is an efficient method for extracting rigidity and flexibility information of biomolecules without performing costly molecular dynamics simulations. The standard algorithm works on a multi-graph associated to a mechanical model constructed from an arbitrary atom-bond network. Motivated by large scale protein flexibility and simulated unfolding applications, we have developed a faster and more robust variation tailored to the specificities of bio-polymers. We describe this new *phased pebble game* as implemented in the new version of our software Kinari-2.**

***Index Terms*—protein rigidity, flexibility, simulated unfolding, dilution, rigid clusters.**

I. Introduction

Large conformational changes are evidence of important functions performed by a protein. Studying such dynamic transitions is however challenging, both from an experimental and computational point of view. *Rigidity analysis* is an alternative method used to extract flexibility information from protein data available in the Protein Data Bank. In this paper we describe a new *phased pebble game* algorithm, engineered for efficiency and robustness of the rigidity calculations implemented in the new version of our software Kinari-2.

Modeling protein flexibility and motion. Hardware enhanced physics-based molecular dynamics simulations have been successful in simulating *fast* protein motions [12]. But such computations are extremely intensive: even with access to large computer clusters or specialized hardware, only sporadic results have been obtained in simulating the motions of a single protein structure at biologically significant time scales. Since *slow* motions can provide significant functional insights about proteins, various alternative methods have been explored.

To achieve efficient and reasonably accurate simulations, there is a growing consensus that detailed, full atom information must be sacrificed in favor of *coarse-grained models,* which partition the molecule's atoms into clusters and treat each cluster as a single unit. Common clustering criteria are based on the chemical structure, distance cut-offs or various manners of modeling the inter-atomic bonds and interactions. As an example of the first kind, protein residues can roughly be represented by their C_α atoms; the disadvantage is that, unless incorporated by other means in the model, residue-specific information is lost. Standard distance-based criteria use a cutoff distance, and approximate the interactions between atoms to shorter range versions.

The Gaussian Network Model (GNM) and its many variations are among popular coarse-grained approximation models, with several implementations widely available (e.g. [22]). They view the interactions between protein residues as elastic springs, inducing oscillatory motions subject to Hooke's Law. An associated matrix is used to compute the normal modes and to group the residues into domains, which help infer large scale displacements among them. Useful functional information can be gathered in this manner [15]. GNM has been shown to have good agreement with experimental crystallographic B-factors. Due to both experimental artifacts and motions within the molecule, it provides a measure of uncertainty for atomic positions in crystal structures solved with X-ray crystallography [13]. GNM calculations have also been used to compute clusters of protein residues, or GNM-based domains, which have been compared to the structural domains assigned by classification schemes (SCOP, CATH) and experimental crystallographers [15].

RigidFinder [1] compares multiple X-ray structures or NMR (Nucleic Magnetic Resonance) models of the same molecule. Closely similar subsets of atoms in the two structures, up to trivial rigid transformations, are treated as rigid clusters, and differences between them are considered flexible regions. By its nature, this method is limited by the availability of NMR data or structures crystallized in multiple conformational states.

Rigidity analysis is an alternative clustering method based on kinematics principles and combinatorial rigidity theory. Instead of using springs to models the molecular bonds and other weak interactions, it treats them as rigid bars or hinges connecting small rigid bodies centered at atoms. Larger groups of atoms that are likely to move together as *rigid* units are then identified. The molecule is decomposed as a collection of rigid clusters connected into a flexible network. This decomposition can then be used as a starting point for analyzing and predicting motions and conformations of a molecule, or

Funded by NIH 1R01GM109456 and NSF CCF-1319366.

for a simplified, combinatorial version of unfolding based on the *dilution of weak, non-covalent interactions*.

The rigid cluster decomposition, illustrated in Fig. 1, provides immediate information on the flexibility of the molecule. For example, having many small clusters in some areas (the uncolored parts in Fig. 1) may be related to small fluctuations responsible for the B-factors. A small number of large clusters (colored differently in Fig. 1) suggests the presence of slow motions that would bring these large clusters apart.

Heuristic implementations of rigidity-based decompositions of protein structures have been available for over 15 years, such as the stand-alone executable ProFlex-FIRST [14], the web server FlexWeb-FIRST [10] and, more recently, our own KinariWeb [7]. Without the need of detailed dynamic simulations, biologically relevant information extracted from rigidity analysis has been demonstrated on a handful of protein structures [9], [11], [18]. Yet, the model has not provided convincing evidence that it has predictive power or that functional information can be *automatically* extracted on a larger scale, in part due to the complexities involved in developing a robust software suite capable to handle very large molecules and datasets. The KINARI project, described next, aims at fulfilling this need.

KINARI (KINematics And RIgidity Analysis of biomolecules) is an on-going project aiming at overcoming many of the limitations of previous rigidity-based implementations. The first version, available since 2011 at http://kinari.cs.umass.edu, is a free open-access web server. In this paper, we present a key algorithmic improvement developed for the software kernel of the second version of KINARI. For comparison purposes, we refer to the first released version of the software [7] as Kinari-1 and use Kinari-2 for the new version described here.

Kinari-1 was designed to analyze a single biomolecule at a time. It was tested and profiled in 2011 on approx. 28000 entries from the Protein Data Bank (PDB); we know that it fails to run to completion on many other entries. This is due primarily to the idiosyncrasies of the PDB file formats and to the limitations in processing very large size molecules. The goal of Kinari-2 is to eliminate these limitations and to make possible the efficient large scale study of protein flexibility. An overview of the new design can be found in [19], and various aspects of the new system have been previously reported in [2]–[4]. In this paper we focus on new methods engineered for allowing rigidity analysis to be performed efficiently on large and very large protein structures and complexes.

Single molecule rigidity analysis, on a large scale. In KINARI, the rigidity analysis of a single molecule proceeds along the computational pipeline described in Fig. 2. The input is a PDB file. The output is a file describing the rigid cluster decomposition of the selected molecule. This file is subsequently sent to a JMol visualizer to produce interactive 3D images with colored clusters as in Fig. 1. A tutorial introduction to the method of rigidity analysis is available on the KinariWeb site [20]. One of the goals of the new design of KINARI was to make it work on a large scale, on big data and on biologically motivated applications. This paper addresses the specific software engineering, modeling, algorithmic, and mathematical problems that underlie the re-designed software kernel based on *pebble game* algorithms.

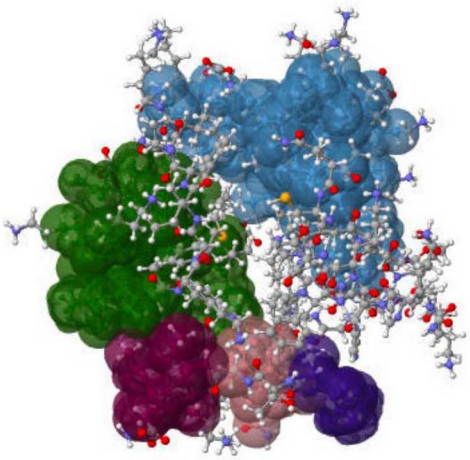

Fig. 1. Rigid cluster decomposition of horse heart cytochrome c (PDB id 1HRC) from [5].

Simulated unfolding with Kinari-Dilution. Bond dilution, an application of rigidity analysis pioneered in [18], is a simplified model of protein unfolding. It starts with a single protein and generates a large dataset of rigid decompositions diluted by strength of non-covalent interactions (hydrogen bonds and hydrophobics). The method was implemented as a prototype in Kinari-1 [6] but, due to an underlying quick-to-implement but slow-to-execute algorithm and an unsatisfactory visualization method for the rigidity dilution results, it was not made publicly available. In Kinari-2 we avoid unnecessary re-runs of the pebble game algorithm by streamlining the kernel code. We also provide an improved visualizer with a dilution-specific consistent colorings of the clusters described in [3], [4].

Overview. This paper focuses on 'methods' and describes the principles underlying the newly engineered phased pebble game for protein rigidity analysis, along with its complexity analysis and implementation in Kinari-2. The algorithm has been tested for correctness and accuracy. The corresponding code and kernel library are currently being integrated in the larger Kinari-2 system. Large scale testing and profiling are under way, and the results are planned to be reported in the long, journal version of this paper.

II. Methods

In this section we start by briefly describing the original pebble game algorithm implemented in Kinari-1, followed by a presentation of the various techniques and optimizations that underlie the phased versions of both the new single-protein and the protein-dilution methods.

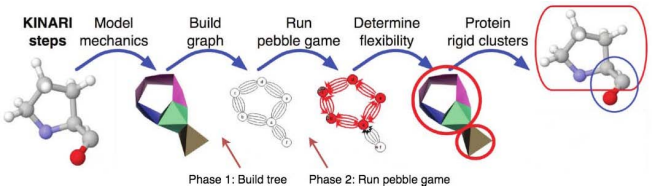

Fig. 2. The processing pipeline of Kinari-1 [7] as modified in Kinari-2. A pre-processing step is extracting the tree for the first part of the phased pebble game, prior to running the second phase.

A. Single Molecule Rigidity Analysis in Kinari.

Kinari rigidity analysis of a single molecule proceeds according to the computational pipeline synthesized in Fig. 2.

The first step is to extract the molecules of interest from the input PDB file, and prepare them for analysis by extracting the atom-covalent bond network and calculating relevant weak bonds and interactions. A mechanical body-bar-hinge structure is constructed from the atom-bond network. A body corresponds to small groups of atoms known a priori to behave like rigid units. For example, a body is assigned to an atom connected by covalent bonds to three or more other atoms, or to an entire peptide unit. Bonds connecting these bodies are modeled as either hinges or with a small number of bars, reflecting their strength. To produce rigidity results consistent with previous implementations, and which were validated on experimental data, it was found that covalent and hydrogen bonds are best modeled as hinges, and hydrophobic interactions as 3 bars. However, this mechanical model has not yet been validated on a large scale. Therefore, Kinari includes a parameter-tuning step, where the user can experiment with other values of the bond modeling parameters, and use a number of bars between 1 and 6 for each type of chemical bond.

The next step is to build a multi-graph representation for body-bar-hinge structure. By using a standard, mathematically proven approach to analyze body-bar-hinge frameworks described in [21], we associate a node to each body, an edge to each bar and 5 edges to a hinge. The resulted multi-graph is the input to the main algorithmic procedure underlying body-bar-hinge rigidity analysis, called the $(6,6)$-pebble game and introduced in [16]. This algorithm is combinatorial and does not depend on the atom coordinates, which makes it much faster than numerical calculations such as the linear algebra required by GNM.

The pebble game algorithm computes a decomposition of the multi-graph into components, which are then translated into rigid clusters in the mechanical and molecular model. Unless the entire structure is rigid, the rigid clusters are connected (through hinges and bars) into a flexible framework. The mechanical interpretation is that any possible motion of the flexible framework *near the native state* maintains these rigid clusters, i.e. the pairwise atomic distances within a given cluster are maintained.

Finally, Kinari integrates a JMol-based visualizer scripted to allow the user to interactively explore the resulting rigid cluster decomposition.

B. The Pebble Game Algorithm

The computational core of Kinari-1 is the provably correct $(6,6)$-pebble game algorithm on multi-graphs from [16]. The algorithm starts by creating a data structure called a *pebbled graph*, which is maintained throughout the execution. The pebble graph is an *oriented* multi-graph with additional data called *pebbles* on its vertices. The initial state of the pebbled graph has 6 pebbles on each vertex of the input multi-graph and no edges.

The algorithm uses an arbitrary ordering of the input edges and loops through them. At each step in the loop, an attempt is made to insert a new edge of the multi-graph into the pebbled graph. The attempt is successful if at least 7 pebbles are present, or can be gathered on the endpoints of the candidate edge; otherwise the edge is rejected. If inserted, the edge is oriented arbitrarily and a pebble is removed from the source of the directed edge. To gather pebbles from other vertices, the rules of the 'game' require a depth-first search along the existing directed edges of the pebbled graph, followed by a 'transport' of the pebble from the location where it was found to its destination via a reversal of direction of edges along the path. After adding all the edges, the remaining pebbles give precise quantitative information on the number of degrees of freedom in the corresponding mechanical structure from which the multi-graph was constructed.

This basic pebble-game algorithm is further enhanced to also compute and maintain the rigid components as sets of vertices in the pebbled graph. When a new edge is inserted, the algorithm checks whether two or more components should be merged together into some new, larger component. *This algorithm lies at the core of the software library underlying Kinari-1.*

C. Phased pebble game algorithm

In Kinari-2, we have developed a variation of the pebble game algorithm with components called the *phased pebble game*. It takes advantage of the fact that polymers such as proteins, RNA and DNA have a specific tree structure with a well-defined linear *backbone*. The underlying tree, properly constructed in a pre-processing phase, is used to improve the algorithm's efficiency by eliminating over $\frac{5}{6}$ of the pebble searches that the original algorithm would have performed.

The phased pebble game splits the edge-inserting loop into two (or more) phases. The state of the pebble game is saved after each phase, and the algorithm is resumed from this saved state rather than started from scratch. For clarity, we illustrate our method only with proteins. Due to the sequential, polymeric structure, it is easy to see that a similar technique can be applied to the DNA or RNA structures available in PDB files.

Initial phase. Instead of starting with an initial empty pebbled graph (with 6 pebbles on all vertices and no edges), the initial phase uses the linearity of the protein backbone and a tree obtained by attaching the residues to it. With the exception of

five residues which have cycles in their atom-bond network (HIS, PHE, PRO, TRP, TYR), the standard amino-acids lead immediately to such a tree structure. The exceptions are treated separately by taking a spanning tree (breaking the cycles arbitrarily). In this phase we use a pre-computed tree produced by a (new to Kinari-2) pre-processing phase which calculates a well structured body-bar-hinge structure from the input PDB file: it is well structured because the order of the bodies and hinges is not arbitrary, but is dictated by this underlying tree structure. The tree has one node for a body and an edge for a hinge arising from the protein covalent structure. The linear backbone has two types of nodes, corresponding, alternately, to the rigid peptide unit and the rigid body centered at the C_α atom. The backbone edges are oriented from the N to the C terminus of the protein. The non-backbone tree edges are oriented *towards the linear backbone*.

Linear Backbone. We do not need to run the pebble game to insert the backbone edges. Rather, we can directly leave the pebbled graph in a state compatible to what would result after inserting all the backbone edges, in their natural, sequential order and oriented from the N to the C terminus of the protein backbone. This state can be generated directly from the input body-bar-hinge file, by placing 1 pebble on each backbone body, except on the C-terminus body, which gets 6 pebbles. It is easy to prove the validity of this simple rule, since the backbone bodies are connected by hinges, and each hinge amounts to 5 edges in the pebbled graph (which thus consumes 5 pebbles and leaves just one on each body except the last one at the C terminus).

Tree towards the backbone. Similarly, the tree edges arising from residue bodies and their covalent bonds, being oriented towards the linear backbone, will lead to the residue bodies all being left with exactly 1 pebble after each edge insertion.

General phased pebble game. Generalizing the previous procedure, our new phased pebble game algorithm saves an internal pebble-graph state and makes it available as the starting state for the next phase. This approach is used to significantly improve the calculation of a protein dilution rigidity analysis, described next.

D. Dilution analysis

Dilution is a simplified model for protein unfolding. It removes all the hydrogen bonds one by one, in the order of some pre-computed energy. Hydrogen-bond dilution [17], [18] is one of the first applications that demonstrated the usefulness of rigidity analysis in protein studies. With the tools provided by the FIRST software [11], the dilution rigidity results are reported using a 1D comparison plot called a dilution plot, which traces the residues inside a rigid cluster with similar colors and along the protein sequence.

In FIRST and in Kinari-1, dilution was performed by running the full rigidity analysis for each structure of the sequential diluted dataset, where one structure differs from the previous one by a single hydrogen bond. The redesigned kernel in Kinari-2 permits the saving of the state of the pebble game on one structure, with the purpose of reusing it as a first-phase when analyzing the rigidity status of the next structure (which has just one more hydrogen bond). Therefore, *we only need to run the pebble game once* and persist the structure each time we add a new bond. When the algorithm is completed with all the edges being inserted, we run a post-processing phase in which the persisted structures are analyzed in order to produce the consistent coloring described in [3], [4] and assign it to all the rigid components, *throughout the dilution*.

III. Implementation

To achieve both high performance and ease of development, we implemented the phased pebble game in C++, and used Python for the pre-processing and post-processing phases.

File formats. The input files are PDB files from the Protein Data Bank. The mechanical structure of the molecule is extracted from a PDB file into an XML-formatted **BBH** (body-bar-hinge) file.

We also have XML-based formats for the multi-graph and the pebbled graph, when it is necessary to have them serialized (e.g. during dilution). Other utilities include conversions from internal pebbled graph format to body-bar-hinge, and this is especially useful during dilution and at the end of a single-molecule rigidity analysis.

Multiplicity of edges. As a simplification in maintaining and using the multi-edges corresponding to different types of bonds, we assign integer weights rather than multiplying the edges. The new pebble game algorithm will attempt to insert a multi-edge rather than a single, simple edge. However, in some cases we will not be able to collect sufficiently many pebbles to insert the entire multi-edge. Instead, our algorithm will add a partial multi-edge (whose multiplicity depends on the number of simple edges that could be accepted), and mark as rejected a corresponding multi-edge with complementary multiplicity. In terms of bonds, this type of acceptance or rejection must be interpreted in terms of rigid redundancy [8], a topic which, for lack of space, is not covered in the current version of this paper. Indeed, a bond must be accepted or rejected as a whole, but its presence inside a rigid component may just make the component *more* rigid, i.e. redundantly rigid rather than minimally rigid. To cover these details and to improve the process of adding edges, we define both a multiplicity and a capacity on the edges of the pebbled graph. The multiplicity is the current accepted weight of the edge (a normal edge has multiplicity of one). The capacity is the maximum possible multiplicity of that edge. As stated above, while trying to add an edge to the pebble graph, we may not be able to add it with the desired multiplicity, i.e. up to full capacity. In preparation for the full treatment of redundancy, we have implemented the algorithm with both the option of rejecting the edge (if not accepted to capacity), or partially accepting it with the currently accepted multiplicity.

Reversed edges. While moving the pebbles in the graph, sometimes we have to reverse the edges through the transferring paths. It is possible that we reverse an edge several times

during the algorithm. Therefore, for each edge in the pebble graph, we add a twin edge with reverse direction, and with the same capacity and zero multiplicity. Therefore, instead of reversing an edge, we just subtract from the multiplicity of the edge and add this multiplicity to its twin.

Overview of the Pebble Game algorithm implementation. The C++ class **PebbleGame** contains the implementation of the original pebble game algorithm from [16]. The algorithm is run on an instance of **PebbledGraph**. This class represents a directed multi-graph and contains a set of graph operations required by the pebble game. It contains methods for DFS, finding connected components and computing the *Reach* of a vertex [16]. A **Component** is represented by a set of vertices in the PebbleGraph. We also maintain a tree structure to preserve the hierarchy of Components created while running the pebble game; this is used for computing the consistent coloring after a dilution calculation. Edges are inserted in the PebbledGraph after collection of pebbles, or rejected and added to a list of rejected edges.

Snapshots of the game. We added the possibility of making single or periodic snapshots of the pebbled graph. The snapshot period, the output format and other parameters are given when we start the game. These snapshots can be used to analyze the changes in the structure of the molecule and its rigidity during a dilution analysis, or, more generally, to resume the algorithm at any particular time.

Phased Pebble Game. Saving the status of the Pebble Game as a pebbled graph makes it is possible to run the game for some part of the input data, save the game status and later resume the game and continue adding other parts of data. In phase one, we insert the backbone, resp. the tree of the protein molecule, and then retain the resulted pebbled graph, exporting it to a file if necessary. Next, we resume the pebble graph for the new bonds, repeating the process for all desired configurations.

Visualization. Kinari-2 integrates a JsMol viewer to visualize the rigidity analysis results. It provides utilities for converting the information in the post-pebble-game BBH file to JMol commands. A consistent coloring for rigid components during dilution is implemented, making the visualization particularly useful for visualizing the process of adding bonds and forming the rigid components during a dilution experiment.

IV. Results

Large scale testing. While designing and implementing the project, one of our main concerns was to insure that our implementation would work correctly and efficiently on large input files, and on a large collection of files. Currently, we have tested our implementation on large randomly generated graphs and are in the process of integrating the large scale testing on large PDB datasets.

Performance improvement. We are performing studies to compare the performance improvements in Kinari-2 over Kinari-1. A thorough report on profiling and testing experiments will be included in the full version of the paper. A back-of-the-envelope analysis, discussed next, provides theoretical evidence that Kinari-2 is expected to perform better than Kinari-1.

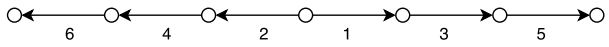

Fig. 3. Linear input graph.

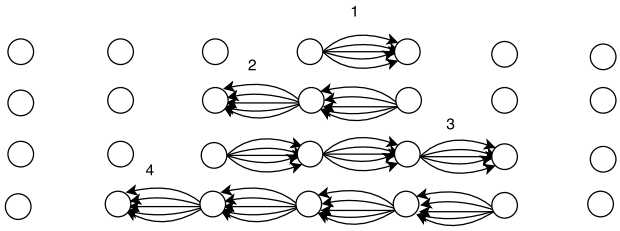

Fig. 4. Pebble game steps in Kinari-1.

Single Molecule Rigidity Analysis. Assume that body-bar-hinge structure of the molecule is given by the graph in Fig. 3 and we want to run pebble game on this graph. In this graph, each node is a body, and each edge is a hinge. The order of adding the hinges is given by the edge numbers of the graph.

In Kinari-1, each hinge is replaced by 5 edges in the multi-graph. It starts with an empty multi-graph and add edges according to the given order. To satisfy the pebble game conditions and collect enough pebbles to be able to add the new edges, the direction of the edges has to be reversed so that the pebbles can be moved (see Fig. 4). In total, it performs $O(5 + (5 + 5) + \ldots + \sum_1^n 5) = O(5n^2)$ edge reversions and pebble movements to add the hinges.

Now let's consider the same input graph in Kinari-2. First of all, by introducing the multi-edges, each hinge will be replaced by a single multi-edge, and this leads to a constant factor improvement in the computations, from 5 to 1. Using the phased pebble game, we can process the backbone of the graph separately, and later add the additional edges to the pebble game. In particular, we can run the pebble game on the backbone in linear time. If the graph has other edges apart from the linear backbone, it would take square time to add them to the pebble graph.

In practice, we can have a backbone of size $n/2$, where n is the size of the molecule. Running the pebble game on these molecules in Kinari-1 would take $O(5n^2)$ time, while in Kinari-2 the running time is $O((n/2)^2 + n/2)$. Although the two approaches have the same asymptotic running times, the difference in their constant factors is noticeable and in practice, this is expected to lead to a significant improvement in performance.

Dilution. Assume the the fixed part of protein has n hinges, and we want to compute the dilution generated by $m << n$ hydrogen bonds. In Kinari-1, this can be done by running the full pebble game algorithm on each structure. As we have

m different structures of sizes n, $n+1$, ..., and $n+m$. the running time of computing the dilution is $O(5n^2+5(n+1)^2+\ldots+5(n+m)^2) = O(5(n^2m+m^2n)) = O(n^3)$.

In Kinari-2, we run the pebble game for the common part of the protein, which is induced by the covalent network. Then for each additional (hydrogen) bond, we add the corresponding edge to the pebble graph and save the state of the graph. The running time is $O(n^2)$, one order of magnitude faster than in Kinari-1.

V. Conclusion

Rigidity analysis is a promising method for protein flexibility analysis, but it requires a large scale validation to demonstrate its full potential. Large scale surveys, e.g. of the entire Protein Data Bank, require improved algorithms and software implementations. In this paper, we presented carefully engineered algorithms designed to take advantage of the specificities of protein structures to improve the performance of Kinari. The theoretical analysis indicates a significant improvement over the previous version of Kinari. A comprehensive testing and profiling of the new implementation is currently under way and the results will be reported in the full version of this paper.

References

[1] A. Abyzov, R. Bjornson, M. Felipe, and M. Gerstein. Rigidfinder: A fast and sensitive method to detect rigid blocks in large macromolecular complexes. *Proteins: Structure, Function, and Bioinformatics*, 78(2):309–324, 2010.

[2] J. C. Bowers, R. T. John, and I. Streinu. Managing reproducible computational experiments with curated proteins in kinari-2. *Proc. Bioinformatics Research and Applications (ISBRA'15), Lecture Notes in Computer Science*, 9096:72–83, 2015. ISBN: 978-3-319-19047-1 (Print) 978-3-319-19048-8 (Online).

[3] E. Flynn and I. Streinu. Consistent visualization of multiple rigid domain decompositions of proteins. *Proc. Bioinformatics Research and Applications (ISBRA'15), Lecture Notes in Computer Science*, 9683:151–162, May 2016.

[4] E. Flynn and I. Streinu. Matching multiple rigid domain decompositions of proteins. *IEEE Transactions on Nanobioscience*, 16(2):1–10, 2017.

[5] N. Fox. Kinari-web case study: 1HRC. Technical report, Smith College, February 2011.

[6] N. Fox. *Accurate and robust mechanical modeling for protein rigidity analysis*. Phd thesis, University of Massachusetts Amherst, 2012.

[7] N. Fox, F. Jagodzinski, Y. Li, and I. Streinu. Kinari-web: A server for protein rigidity analysis. *Nucleic Acids Research*, 39(Web Server Issue):W177–W183, 2011. doi:10.1093/nar/gkr482.

[8] N. Fox and I. Streinu. *Redundant and Critical Noncovalent Interactions in Protein Rigid Cluster Analysis*, pages 167–196. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[9] B. M. Hespenheide, D. J. Jacobs, and M. F. Thorpe. Structural rigidity in the capsid assembly of cowpea chlorotic mottle virus. *Journal of Physics: Condensed Matter*, 16(44):S5055, 2004.

[10] B. M. Hespenheide and M. F. Thorpe. *Flexweb*. http://flexweb.asu.edu.

[11] D. J. Jacobs, A. Rader, L. A. Kuhn, and M. Thorpe. Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Bioinformatics*, 44(2):150–165, 2001.

[12] J. L. Klepeis, K. Lindorff-Larsen, R. O. Dror, and D. E. Shaw. Long-timescale molecular dynamics simulations of protein structure and function. *Current Opinion in Structural Biology*, 19(2):120 – 127, 2009. Theory and simulation / Macromolecular assemblages.

[13] D. A. Kondrashov, Q. Cui, and G. N. Phillips. Optimization and evaluation of a coarse-grained model of protein motion using x-ray crystal data. *Biophysical journal*, 91(8):2760–2767, 2006.

[14] L. A. Kuhn. *First installation and user's guide (floppy inclusion and rigid substructure topography, version 4.0)*, 2004. http://www.kuhnlab.bmb.msu.edu/software/proflex/index.html.

[15] S. Kundu, D. C. Sorensen, and G. N. Phillips. Automatic domain decomposition of proteins by a gaussian network model. *Proteins: Structure, Function, and Bioinformatics*, 57(4):725–733, 2004.

[16] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425 – 1437, 2008. Third European Conference on Combinatorics – Graph Theory and Applications.

[17] A. Rader, G. Anderson, B. Isin, H. G. Khorana, I. Bahar, and J. Klein-Seetharaman. Identification of core amino acids stabilizing rhodopsin. *Proceedings of the National Academy of Sciences of the United States of America*, 101(19):7246–7251, 2004.

[18] A. Rader, B. M. Hespenheide, L. A. Kuhn, and M. F. Thorpe. Protein unfolding: rigidity lost. *Proceedings of the National Academy of Sciences*, 99(6):3540–3545, 2002.

[19] I. Streinu. Large scale rigidity-based flexibility analysis of biomolecules. *Structural Dynamics*, 3(012005), January 2016.

[20] I. Streinu, F. Jagodzinski, and N. Fox. Tutorial: Analyzing protein flexibility: an introduction to combinatorial rigidity methods and applications. In *IEEE International Conference Bioinformatics and Biomedicine (BIBM 2011), Atlanta, GA, Nov 12-15, 2011*, 2011.

[21] T.-S. Tay. Rigidity of multi-graphs. i. linking rigid bodies in n-space. *J. Comb. Theory, Ser. B*, 36:95–112, 1984.

[22] L.-W. Yang, A. J. Rader, X. Liu, C. J. Jursa, S. C. Chen, H. A. Karimi, and I. Bahar. ognm: online computation of structural dynamics using the gaussian network model. *Nucleic Acids Research*, 34:W24–W31, 2006.