HIGH DIMENSIONAL INDEXING FOR PROTEIN STRUCTURE MATCHING USING BOWTIES

ZI H HUANG, XIAOFANG ZHOU
School of Information Technology and Electrical Engineering
ARC Centre in Bioinformatics
The University of Queensland, QLD 4072 Australia
{huang, zxf}@itee.uq.edu.au

DAWEI SONG
CRC for Distributed Systems Technology
The University of Queensland QLD 4072 Australia
dsong@dstc.edu.au

For determining functionality dependencies between two proteins, both represented as 3D structures, it is an essential condition that they have a matching structure. As 3D structures for proteins are large, complex and constantly evolving, it is very time-consuming to identify possible locations and sizes of such a matching structure for a given protein against a large protein database. In this paper, we introduce a novel representation model and apply a transformation and formalization to this problem. We then propose a database solution by using innovative high dimensional indexing mechanisms. Experimental results demonstrate a promising performance of the high dimensional indexing to this biologically critical but previously computationally prohibitive problem.

1. Introduction

The structure of a protein can be represented as a collection of points (atoms) or vectors (from one atom to another) in a three dimensional space. It has been shown that *Protein properties are a direct consequence of the protein's unique three-dimensional structure.*[11] Certain structural regions of a protein often perform some specific function. Analyzing the three-dimensional structure of a protein therefore provides a basis for understanding its biological functionality. Having a matching (similar) structure has been considered as an essential condition for the existence of potential interaction between two proteins. As 3D structures for proteins are large, complex and constantly evolving, it is very time-consuming to identify possible locations and sizes of such a matching structure for a given protein against a large protein database. In this paper, we adopt a novel vector representation and formalize the protein structure matching problem. We propose a database solution and investigate various innovative indexing mechanisms. Our initial experimental results demonstrate a promising performance to this biologically critical but previously computationally prohibitive problem.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to protein structures as well as an overview of related work. Section 3 defines the 3D vector representation of protein structures and formalizes the protein structure matching problem. A database solution to the problem is proposed in Section 4. Various high dimensional indexing approaches to facilitate efficient structure matching are investigated in Section 5. Section 6 shows the experimental results. Section 7 concludes the paper and highlights the future work.

2. Preliminaries and Related Work

A protein is a large molecule composed of one or more chains of amino acids in specific order. Each amino acid contains a central atom C_α to which a *sidechain* R, an amino (N-H) group and a Carboxyl ($C' = O$) group are attached. For each of these amino acids except *Glycine* (which is the simplest amino acid without a sidechain), the *sidechain* is connected to C_α via another atom C_β.[4] A protein is constructed from amino acids that are linked by peptide bonds forming a polypeptide chain(Figure1).

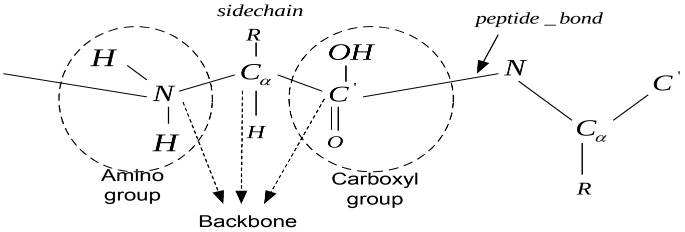

Figure 1. Structure of amino acid.

The amino acid sequence of a protein's polypeptide chain is called its primary structure, which can be represented a linear string of residues, abbreviated with one-letter codes.

Different regions on the sequence form regular secondary structures, including the α helices and β sheets in a three-dimensional space. As a consequence, the protein's structure can then be folded into a three-dimensional configuration.

2.1. *Structure and sequence similarities*

It is important to distinguish structural and sequence similarities, according to which proteins can be classified. The former is an indicator of an evolutionary relationship between linear sequences, while the latter is based on comparison between atoms or regions in three dimensional space. *Significant structural similarity is common, even among proteins that do not share any sequence similarity or evolutionary relationship.*[12] Structural comparison and alignment service can discover similar patches from two proteins without any measurable sequence similarity. This paper will be focusing on the structure similarity.

2.2. *Methods for comparing of 3D protein structures*

The protein structure can always be represented as a set of points (atoms) in 3D space. For example, PDB (Protein Data Bank)[1] uses this method by arranging a protein on an imaginary Cartesian coordinate frame and assign (x,y,z) coordinates to each atom. This representation serves as a basis of different simplified high level representations.

Distance Matrix A matrix of inter-atomic distances can be constructed to represent the three-dimensional structure. Atom-atom distance (e.g., C_α-C_α, C_β-C_β) matrix has been used by many similar structure searching approaches.

The SSAP system[14] uses a distance plot-based method to compare internal geometry between proteins via the Needleman - Wunsch dynamic programming algorithm.[13] The structure of a protein is represented by describing a structural environment for each amino acid as a set of vectors from its C_β atom to C_β atoms of all the other amino acids in the protein. If the structural environments in two protein structures are similar, the structures are supposed to be similar.

The DALI system[8,9,10] also uses distance matrix method to compare structural relationships between the proteins. The residue-residue (C_α-C_α) distance matrix is calculated. Distance plot-based methods compare all the inter-residue distances in one protein to corresponding distances in another. Similar patches of residues in two proteins are superimposed as closely as possible into a common core structure by minimizing the sum of the atomic distances between the aligned C_α atoms.

The SARF system[2] performs comparison between protein structures on the level of secondary structures, represented as vectors of C_β-C_β atoms, instead of residues. It searches large sets of secondary structure elements in two protein structures which could be superimposed with a small RMSD (Root Mean Square Deviation). Another system, VAST,[7] adopts the similar representation.

Abstract Chain Fold It is a usual way to tabulate the torsion (dihedral) angles for each residue to reconstruct protein structure using standard covalent bond length and angles.[3] The backbone can be further represented by virtual bond between C_α-atoms. The main-chain is fully described by the virtual dihedral angle α_i defined by four successive C_α atoms and the virtual bond angle. This description is used for building backbone wire models.

Our Approach All the above mentioned methods are based on measuring internal C_α-C_α (residue-residue) or C_β-C_β (sidechain-sidechain) distances. This paper will adopt a different way by representing a protein's structure as vectors of C_α-C_β atoms, which has been advocated by Mckie 1995.[5] The overall spatial relationships between the C_α-C_β vectors will be taken into account.

3. Problem Formulation

3.1. *Vector representation of a protein in 3D space*

Since positions of C_α atoms are variant from time to time, C_β atoms are introduced in our method to give more information for representing a protein's structure(Figure 2). A pair of C_α-C_β atoms in the same residue constructs a vector from C_α atom to C_β atom. A protein (or more precisely, a snapshot of a protein, as the shape of a protein can change over time) can be defined as a vector collection:

$$P = \{v_i | 1 \leq i \leq n\} \tag{1}$$

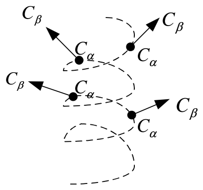

Figure 2. Protein structure as 3D vectors.

Each v_i is a vector of (C_α, C_β) for residue i, n is the number of residues (except Gly) of a protein. The number of vectors in a protein can vary between 10 to 10,000. The length of a vector (i.e., from its α end to the β end) is always about 1.5 Å (angstrom).

Since PDB (Protein Data Bank) supplies the coordinates of each atom of proteins, it is easy to build a $C_\alpha C_\beta$ vector space and represent a protein as a collection of vectors in a three-dimensional space.

3.2. *Bowties*

For two vectors u and v, their spatial relationship can be described using four distances between their C_α and C_β ends. We denote these four distances, as illustrated in Figure 3 (a), as $d_{\alpha\alpha}$, $d_{\beta\beta}$, $d_{\alpha\beta}$, and $d_{\beta\alpha}$. All these four distances are Euclidean distances in 3D space. This characterization of spatial relationship between two vectors using the four distances is called the "bowtie" method. A bowtie consisting of vectors u and v is denoted as $B_{u,v}(d_{\alpha\alpha}, d_{\beta\beta}, d_{\alpha\beta}, d_{\beta\alpha})$, in short $B_{u,v}$.

As a vector is directional and can point to any direction in a 3D space, twisting a bowtie $B_{u,v} = (d_{\alpha\alpha}, d_{\beta\beta}, d_{\alpha\beta}, d_{\beta\alpha})$ in the 3D space leads to $B_{v,u} = (d_{\alpha\alpha}, d_{\beta\beta}, d_{\beta\alpha}, d_{\alpha\beta})$, which is considered identical to $B_{u,v}$. If the two diagonal distances $d_{\alpha\beta}$ and $d_{\beta\alpha}$ are ordered, we can represent $B_{u,v}$ and $B_{v,u}$ in an unified form $B(d_{\alpha\alpha}, d_{\beta\beta}, min(d_{\alpha\beta}, d_{\beta\alpha}), max(d_{\alpha\beta}, d_{\beta\alpha}))$.

It is obvious that the maximal number of bowties derived from n vectors is C_n^2. A bowtie whose $d_{\alpha\alpha}(u, v)$ distance is not greater than 25Å is referred to as a *qualified bowtie*. The constraint of 25Å here reflects the distance cut-off of a distant contact in a protein structure.

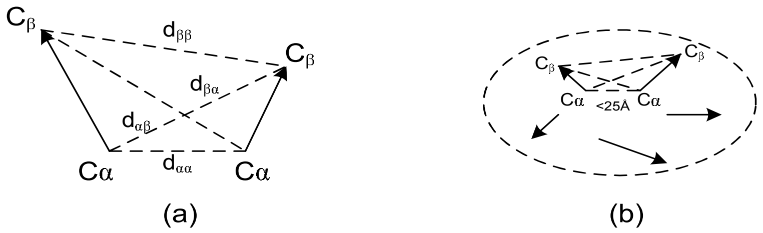

Figure 3. (a)The bowtie of vectors u and v (b)An example of motif

For any two bowties B and B', they are similar if the following conditions are satisfied:

(1) $B(d_{\alpha\alpha}) \approx B'(d_{\alpha\alpha}) \wedge B(d_{\beta\beta}) \approx B'(d_{\beta\beta})$, and
(2) $min(B(d_{\alpha\beta}), B(d_{\beta\alpha})) \approx min(B'(d_{\alpha\beta}), B'(d_{\beta\alpha})) \wedge$
$max(B(d_{\alpha\beta}), B(d_{\beta\alpha})) \approx max(B'(d_{\alpha\beta}), B'(d_{\beta\alpha}))$

The second condition considers bowtie similarity by allowing mirror images. Clearly, if all bowties are stored in the unified form, this condition can be simplified. The relationship ”$\approx$” reads as ”equals to within a tolerance”, and the tolerance is typically 1.5Å.

3.3. *Motifs*

The structural regions of a protein can be modeled by motifs. A motif M of a protein P is defined as a subset of vectors:

$$\begin{aligned} M = v_{i1}, v_{i2}, ..., v_{im} \subseteq P \\ \forall(v_{ik}, v_{il}) \in M, d_{\alpha\alpha}(v_{ik}, v_{il}) \leq 25\mathring{A} \end{aligned} \tag{2}$$

Obviously, a motif can also be viewed as a set of qualified bowties, formed by all the possible vectors pairs whose α-α distance is less than 25Å.

As an illustration, Figure 3 (b) visualizes a motif:

3.4. *Motif Matching*

For two motifs $M_1 = \{q_1, ..., q_m\}$ and $M_2 = \{r_1, ..., r_n\}$, they are similar if there exists a sub-motif $S_1 \subseteq M_1$ and a sub-motif $S_2 \subseteq M_2$, $\forall(v, v') \in S_1 \times S_1$, $\exists(u, u') \in S_2 \times S_2$, such that $B(v, v') \approx B(u, u')$ and the sizes of S_1 and S_2 are larger than 5 but smaller than 20.

3.5. *Protein structure matching*

For two proteins P_1 and P_2, they have a matching patch if there exists motif $M_1 \subseteq P_1$ and a motif $M_2 \subseteq P_2$ such that $M_1 \approx M_2$. In summary, given a query protein Q, the problem we investigate is to find all the proteins from a protein database such that the resultant proteins have a one or more matching motifs with Q.

4. A Database Solution to the Problem

The protein structure matching problem can be split to three tasks, which are construction of 3D vector database - extracting $C_\alpha C_\beta$ vectors from the PDB via pre-processing; motif detection and indexing; and matching. The matched proteins or fragments of proteins (motifs) will be returned for post-processing of functional analysis, which is out of the scope of this paper. The key issue is an indexing schema to facilitate efficient searching.

As defined in Section 3, a motif can be viewed as a set of bowties and the matching of two motifs is done via the one-to-one mapping of similar bowties between the two maximally matched sub-motifs. Therefore, efficient indexing of bowties is essential to the motif formation and matching. The following sections will focus on this issue.

5. Indexing and Querying Bowties

The most commonly used one-dimensional indexing approaches in the database literature are *hashing* and the B$^+$-tree. The hash based method basically uses a hashing function to map search key values into a range of bucket numbers. However, the hashing method does not support range queries, which are exactly what we need to match two bowties within the 1.5Å tolerance. Therefore, the hash indexing is not applicable to the problem we deal with in this paper.

The B$^+$-tree maintains a dynamic index structure, which is a balanced tree where search is directed by its internal nodes (index entries) and data entries are stored in its leaf nodes. An advantage of B$^+$-tree indexing is that it provides efficient support to the range queries without decreasing the efficiency of equality selections. Therefore, only B$^+$-tree based (and more generally, tree-based) indexing methods will be considered in this paper.

Recall a bowtie is represented by four distances. Thus bowties can also be viewed as points in a four dimensional space. The most popular multidimensional (spatial) access method is R-tree indexing, which has been provided in most commercial database management systems such as Oracle.

The R-tree is a height-balanced data structure like B$^+$-tree. It is based on the approximation of a complex spatial object (or a group of spatial objects) with the minimum bounding rectangle (*MBR*) that encloses the geometry. The sides of a MBR are parallel to the axes of the data space. An R-tree consists of a hierarchical index on the MBRs of the geometries. For illustration, Gaede's paper[6] shows an R-tree for a working example. Because R-tree indexing is fast and works directly on geodetic data, it has been widely used for working with spatial data. The R-tree idea is applicable to higher dimensional data indexing, and high dimensional R-trees are supported by Oracle and many other database management systems.

Using Four B$^+$-Trees A simple way to index bowties is to create four B$^+$-tree indexes separately on the four distances of a bowtie. Given a query bowtie, four separate searches are conducted. The final result will be the intersection of the four sets of intermediate results. This approach may generate up to four immediate datasets (using one B$^+$-tree for selection using one distance). It is clear that the selectivity using one distance is much higher than that using four distances together, such intermediate datasets can be very large. These large intermediate results are costly to generate and store, and costly to merge, and have no indexes on them. It is clear that such one-dimensional indexes are not really suitable for supporting multi-dimensional queries.

Using One 4D B$^+$-Tree Instead of building a separate B$^+$-tree on each distances, a bowtie is considered as a quaternary set of its four distances ordered by descending priorities: $d_{\alpha\alpha}$, $d_{\beta\beta}$, $d_{\alpha\beta}$, and $d_{\beta\alpha}$. The "quartets" are then indexed by using a B$^+$-tree, wherein each key stores an ordered quartet. We refer this as the 4D B$^+$-tree approach, which is considered here a "pseudo" high-dimensional indexing. A similar approach was proposed by Wang 2002[15] for pattern discovery in a three dimensional space.

Using One 4D R-Tree By considering the four distances as coordinates, a bowtie can be mapped to a (x, y, z, k) point in the four dimensional space. The bowties (as 4D points) can be treated as spatial objects and then indexed using a 4D R-tree. Given a query bowtie Q = (x_1, y_1, z_1, k_1) and a tolerance value ε. The R-tree query can be represented as a 4D cube $(x_1 \pm \varepsilon, y_1 \pm \varepsilon, z_1 \pm \varepsilon, k_1 \pm \varepsilon)$, with Q as its centroid. In our case, ε is the tolerance value 1.5Å.

Using 2D R-Trees The four distances of a bowtie can be grouped into two 2D points $(d_{\alpha\alpha}, d_{\beta\beta})$ and $(d_{\alpha\beta}, d_{\beta\alpha})$. Then two 2D R-trees can be used to index the set of first points and the set of second points separately. Given a query bowtie Q = $((x_1, y_1), (z_1, k_1))$. Its R-tree query consists of two 2D rectangles $(x_1 \pm \varepsilon, y_1 \pm \varepsilon)$ and $(z_1 \pm \varepsilon, k_1 \pm \varepsilon)$, where ε is the tolerance value 1.5Å.

Using One 3D R-Tree We can also consider a bowtie as a 3D point $(d_{\alpha\alpha}, d_{\beta\beta}, d_{\alpha\beta} + d_{\beta\alpha})$. Given a query bowtie Q = $(x_1, y_1, z_1 + k_1)$ and a tolerance value ε. The 3D R-tree query can be represented as a 3D cube $(x_1 \pm \varepsilon, y_1 \pm \varepsilon, z_1 + k_1 \pm 2\varepsilon)$, with Q as its centroid.

6. Bowtie Indexing Experiments

6.1. *Test Data*

Over 20,000 proteins (13.5G) protein data (in format of `mmcif`) are downloaded from Protein Data Bank.[1] A total of 448 sample proteins are randomly selected for our initial experiments. For the sample protein data set, the average number of vectors for each protein is 174. The total size of final vector space is 78,218, from which 5,272,573 qualified bowties are built. Oracle 10g with Spatial Data Option is used to store all bowtie data, to create both B-tree and 2-4 dimensional R-trees, and to process all queries (represented in SQL). No special code is used for bowtie similarity search.

6.2. *Queries*

A set of three query bowties $Q_{max} = (24.9, 28, 26.5, 26.5)$, $Q_{min} = (2.6, 2.7, 2.9, 2.6)$, $Q_{avg} = (17, 17.2, 17.1, 17.1)$ is used throughout the experiment. They are selected to cover the cases of maximal, minimal and average values of the four distances in the test data. The numbers of matching bowties are respectively 34530, 104, and 456520, indicating the query bowties differ from each other in density of data distribution in the query boxes.

The query bowties and the 1.5Å tolerance are translated to different SQL queries with respect to different indexing mechanisms under investigation. The follow example shows the Oracle SQL queries of Q_{avg} for 4D R-tree indexing:

```
query_box3 mdsys.sdo_geometry(4003, null, null,mdsys.sdo_elem_info_array(1,1003,3),
        mdsys.sdo_ordinate_array(15.5, 15.7, 15.6, 15.6, 18.5, 18.7, 18.6, 18.6));

select * from h_4d_test where mdsys.sdo_filter( h_4d_test.point, query_box3,
       'querytype=window layer_gtype=POINT')='TRUE';
```

Note that the query bowtie is represented as a 4D cube and its bottom-left and top-right corners are used to identify the geometry.

6.3. *Performance Indicators*

Disk reads (number of times the disk is read), *Buffer Gets* (number of times the buffer in main memory is read) and *CPU time* (in seconds for SQL query parsing and executing, and query result fetching) are chosen to measure the efficiency of the indexing methods. The first two are considered as indicators of intermediate data set traversed during the query processing. In addition, precision (the percentage of the returned bowties being correct) and recall (the percentage of correctly matching bowties being returned) are used as effectiveness measures.

6.4. *Experimental Results*

Figure 4 summarizes the experimental results. High dimensional indexing, as expected, generally saves CPU time and reduces intermediate Dataset sizes (disk reads and buffer gets), in comparison to using four separate one dimensional indexes. Scanning four B$^+$-trees produce four sets of intermediate data which may contain a large amount of duplicated bowties. Joining the fours sets to find their intersection leads to more CPU time used.

The 4D B$^+$-tree demonstrates a quicker response time and less intermediate data by combining the four distances into a single key which has less unique values and thus less number of intermediate nodes in the tree.

The 2D R-trees approach requires the least disk reads and buffer gets, suggesting the spatial indexing in 2D space does help reduce the intermediate data.

However, 3D and 4D R-Trees are not as good as 2D R-Trees. This is probably due to the fact that already skewed data will be more skewed in higher dimensional spaces. Therefore, there is a high degree of overlapping among MBRs containing the 3D and 4D points, resulting in a large amount of overlapping with the query box and in turn a large number of subtrees are traversed. Particularly for the 3D method, the query box is bigger than others since its third dimension is the sum of the $d_{\alpha\beta}$ and $d_{\beta\alpha}$ so that the tolerance value for this dimension is doubled.

	CPU TIME	DISK READS	BUFFER GETS	PRECISION	RECALL
4×B+ Tree	25.2	203314	201347	100%	100%
4D B+ TREE	1.6	35851	35872	100%	100%
2×2D R-TREE	1.1	1681	12520	100%	100%
3D R-TREE	1.4	34948	38692	99.7%	100%
4D R-TREE	2.4	42414	45646	84%	100%

(a) Experimental results of query Q_{max}.

	CPU TIME	DISK READS	BUFFER GETS	PRECISION	RECALL
4×B+ TREE	21.9	407370	168770	100%	100%
4D B+ TREE	1.3	35849	35862	100%	100%
2×2D R-TREE	0.03	178	1004	100%	74%
3D R-TREE	0.02	303	669	46%	100%
4D R-TREE	0.04	209	447	61%	80%

(b) Experimental results of query Q_{min}.

	CPU TIME	DISK READS	BUFFER GETS	PRECISION	RECALL
4×B+ Tree	42.4	313002	323085	100%	100%
4D B+ TREE	1.7	35849	35862	100%	100%
2×2D R-TREE	11.2	24869	123828	100%	100%
3D R-TREE	40.5	616707	669350	75%	100%
4D R-TREE	22.6	531884	565838	88%	100%

(c) Experimental results of query Q_{avg}.

Figure 4. Experimental results.

It can also be observed that the 4 B$^+$-trees approach and one 4D B$^+$-tree approach achieve 100% precision and recall, due to their nature of using exact values of attributes as search keys. On the other hand, all the R-Tree based methods cannot guarantee this, as they are based on approximations. The 2D method achieves much higher precision and recall than the other two due to the same reasons discussed above for intermediate set. This can also be explained by considering data skew and the trend of more skewed data when the number of dimensions increases.

It seems the performance of each approach is dependent on the number of correctly matching bowties of different test queries. For a query with lower such number, e.g., Q_{min} (104), the 2D R-Tree would seem the best. For query with higher such number, such as Q_{avg} (456,520), the 4D B$^+$-tree shows its advantage in both CPU time and Buffer gets.

7. Conclusions and Future Work

We have formulated a problem of protein structure matching in a 3D space of C_α-C_β vectors. The problem is broken down into finding matching similar motifs. A motif can be viewed as a set of bowties and the matching of two motifs is done via the one-to-one mapping of similar bowties between the two maximally matched sub-motifs. Therefore, efficient indexing of bowties is essential to the motif formation and matching. As a bowtie is represented by four distances describing the spatial relationship between two vectors, it is hypothesized that high dimensional indexing would be more appropriate for this purpose. We have investigated in detail various high-dimensional indexing approaches and compare their efficiency and effectiveness using a collection of over five millions bowties derived from 448 proteins. The experimental results demonstrate the advantages of high-dimensional indexing over one-dimensional approach. The 2D R-tree and 4D B^+-tree approaches are observed as the best performing ones. However, this needs to be further verified using larger scale data set. We leave it as part of our future work.

The work we have done so far has been focusing on the bowties level. In the future, we need to move up to the motif level for an efficient motif detection and matching algorithm, which will be underpinned by bowtie indexing and matching mechanisms reported in this paper.

Acknowledgements

The work reported in this paper has been funded in part by the Australian Research Council (Grant No. DP0344488) and the Co-operative Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Education, Science and Training). The authors would like to thank Sham Prasher, Lynn Teo, Mark Smythe, Gerald Hartig and Richard Cole for their kind assistance.

References

1. Protein data bank. http://www.rcsb.org/pdb/.
2. N.N. Alexandrov and D. Fischer. Analysis of topological and nontopological structural similarities in the pdb: New examples with old structures. *Proteins*, 25:354–365, 1996.
3. T.L. Blundell and M.S. Johnson. Catching a common fold. *Protein Sci.*, 2:817–833, 1993.
4. C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, Inc., 1998.
5. Mckie et al. In *Peptides: Chemistry, Structure and Biology*, pages 354–355, 1995.
6. V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
7. J.-F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.*, 6:377–385, 1996.
8. L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233:123–138, 1993.
9. L. Holm and C. Sander. Searching protein structure database has come of age. *Proteins*, 19:165–173, 1994.
10. L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–603, 1996.
11. A.E. Mirsky and L. Pauling. On the structure of native, denatured, and coagulated proteins. *Proceedings of National Academy of Sciences of the United States of America*, 22:439–447, 1936.
12. D.W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2001.
13. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.
14. C.A. Orengo and W.R. Taylor. Ssap: Sequential structure alignment program for protein structure comparison. *Methods Enzymol.*, 266:617–635, 1996.
15. X. Wang. $\triangle b+$ tree: Indexing 3d poing sets for pattern discovery. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 701–704, 2002.