

1 Voting scheme with hash table

Many-to-many comparisons are evaluated when we align protein structures. In order to avoid repetition, a better organization of computation is necessary. This could be achieved by pre-computing the indexes of proteins and arranging them in a hash table. Then, queries are evaluated based on a voting scheme using the hash table. This voting scheme replaces the seed generation process.

In this lecture, we look into the voting scheme used in 3dSEARCH [2]. The algorithm is based on the concept of geometric hashing [1] developed in the field of computer vision. The basic idea is to represent all secondary structure elements (SSEs) from all target proteins with a large, highly redundant hash (or index) table. Once the table has been constructed, every SSE from a given query structure can be compared simultaneously to the entire set of SSEs of the target structures, by indexing the SSE into the table.

The hash table that consists of 3-dimensional regular grid of cube bins (2\AA) is constructed as follows. For each target structure, we compute a coordinate system P for every pair of vectors (i, j) (Figure 1). Then, we transform all

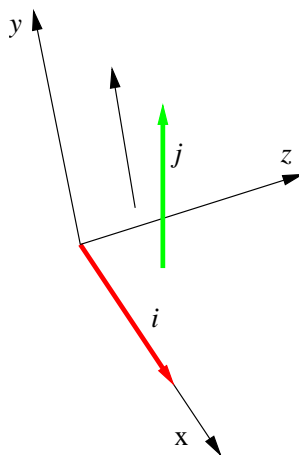


Figure 1: The coordinate systems for the vectors (i, j) . The z -axis is coincident with i and the y -axis is parallel to j .

remaining vectors in the protein to the coordinate system P . At last, for each remaining vector k , we place an entry (with the orientation, coordinate system and type of k) into the hash table at the location specified by the coordinates of the mid-point of k (Figure 2).

Since the grid is sparsely occupied, so does the hash table. A structure with n SSEs contributes $n(n-1)(n-2)$ entries. Each vector is represented $(n-1)(n-2)$ times. For instance, 10000 structures with 10 SSEs each yield around 7M entries.

Now, we can process any query structures using this pre-computed hash table. Given a query structure Q , we compute a coordinate for each pair of vectors (i, j) of Q . Next, for each other vector k , we retrieve the bin accessed by this vector and the neighboring bins. For every vector contained in those bins that has the same orientation and type (helix, strand) as k , a vote is added for the coordinate system of the corresponding target structure. We sort the target structures based on the maximum number of votes it received by any of its coordinate systems. Now, we can find a better alignment using LOCK. Since the set of target structures is small, the execution of LOCK takes seconds, compared with hours for pure LOCK.

Voting systems have the following advantages. Firstly, they are very efficient in practice for many-to-many comparisons. Secondly, they can establish correspondence between partial, disconnected sub-structures. Thirdly, parallel implementation is straightforward. Lastly, they are independent of the order in which the vectors are considered.

However, the voting systems might establish correspondences that do not satisfy the backbone sequence constraints. If we want to keep the constraint satisfied, we will have to ensure votes are only given to the vectors that satisfy the constraints. This introduces extra computation.

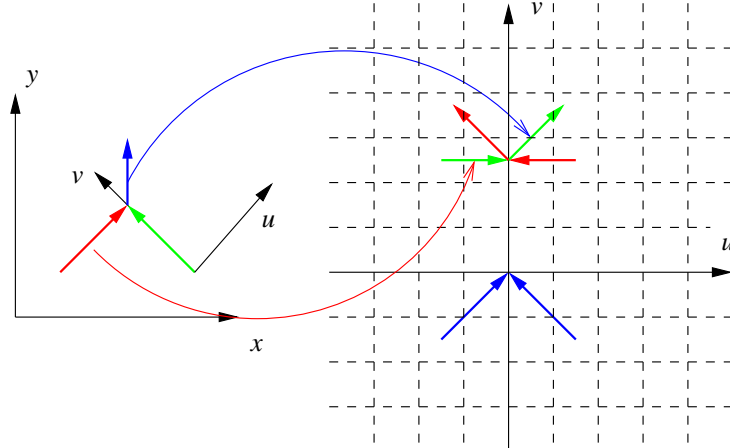


Figure 2: Consider a 2-dimensional case. Let the (original) vector space form the x, y -plane in the left-handed side of the figure, and (u, v) be the new coordinate system based on the green vector. The hash table is given in the right-handed side of the figure. Now, the (remaining) blue and the red vectors in the original vector space are, based on their mid-points, mapped to the green vectors stored in the hash table. Similarly, when the coordinate system is created based on the red (blue) vector in the original vector space, the other two vectors will be mapped to the red (blue) vectors in the hash table. Notice that each vectors is represented twice in the hash table.

2 Finding Pharmacophore in Ligands

A ligand is a molecule that binds to another molecule to form a larger compound. For proteins, this can have the effect of inhibiting the proteins function or catalyzing its activities. Given a collection (5-10) of small flexible ligands with similar activity (binding at same sites), we would like to find a substructure common to all the ligands (a pharmacophore).

A pharmacophore is a specific, three dimensional map of biological properties common to all active low-energy conformations of a set of ligands which exhibit a particular activity (see Figure 3). Conceptually, a pharmacophore is a distillation of the functional attributes of ligands which accomplish a specific task.

2.1 Pharmacophore and Rational Drug Design

Pharmacophores are conceptual templates for drug design. Pharmacophore identification is a form of “reverse engineering” to get a model of a binding site. Once it is extracted from a set of ligands, a pharmacophore can be used as a model for the design of other molecules that can accomplish the same activity. A pharmacophore can also be used to modify ligands into more potent drugs and/or to screen large databases of ligands for “leads”.

Computational tools have greatly enhanced the drug design process in recent years. Here are some software used for searching pharmacophore:

- DISCO [Martin et al., 1993]
- DISCOtech and GASP [Tripos, Inc.]
- CATALYST and HIPHOP [Accelrys et al.; Green et al., 1994; Barnum et al., 1996]
- RAPID [P.W. Finn et al., 1998]

Next, we shall discuss how RAPID work in detail.

2.2 RAPID: Randomized Pharmacophore Identification for Drug Design

RAPID is an integrated software system design to address the pharmacophore identification problem:

Given a set of n ligands, $\{M_1, \dots, M_N\}$, that interact with the same receptor, find geometric invariants of these ligands, i.e., a set of features embedded in R^3 that is present in one or more valid conformations of each ligands. Note that these set of features are representatives of pharmacophore candidates.

In RAPID, the two main modules are conformational search and invariant identification. Conformational search will produce a set of distinct low-energy conformations, $\{C_{i1}, C_{i2}, \dots, C_{ik}\}$, for each of the n ligand. In the invariant identification module, there are two phases, namely Pair-Probe and Multi-Probe.

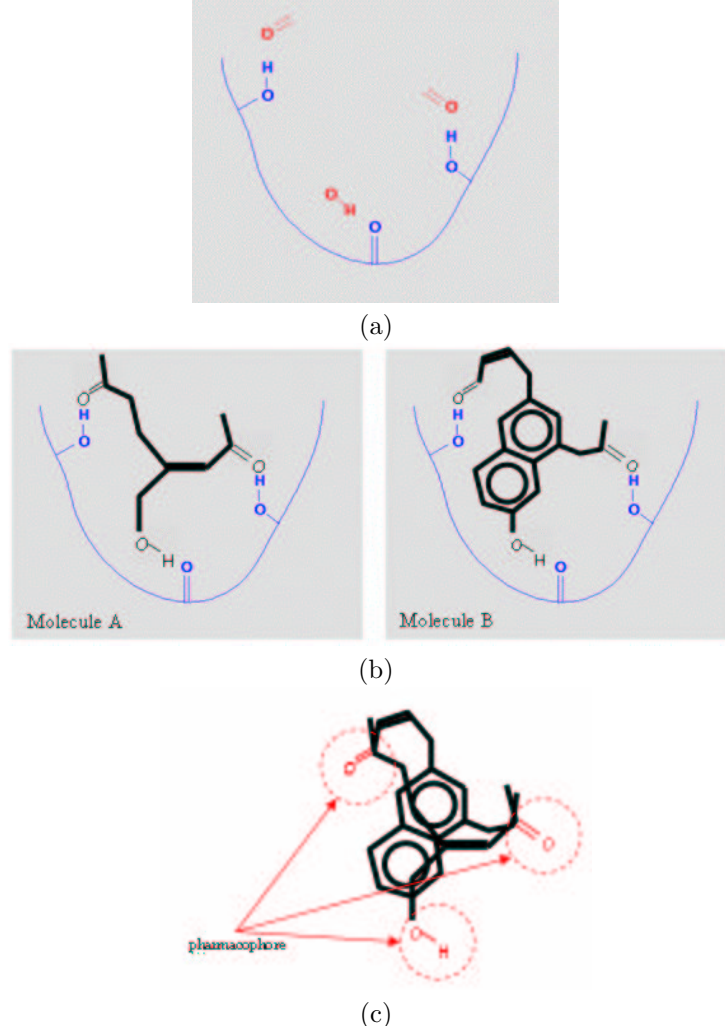


Figure 3: (a) Model of a binding site, (b) binding of two molecules A and B, (c) pharmacophore of A and B

Given the smallest number n of atoms/features in a ligand, some constant a ($0 < a \leq 1$) and two conformations P_1 and P_2 of two distinct ligands (or candidate invariant, obtained from conformational search module), Pair-Probe(P_1, P_2) repeats the following steps s times:

1. Pick a triplet of atoms at random from P_1
2. Determine three atoms in P_2 congruent to this triplet; compute the alignment transform T
3. Iterate: Apply T to P_2 ; determine the atoms in P_1 matching those in P_2 ; update T
4. If the number of matching atoms exceed an , then return this atom set as a candidate invariant S

We can calculate the magnitude of s as follows:

- Probability(picking 3 atoms in invariant) $\approx a^3$
- Probability(failing to find invariant) $\approx (1 - a^3)^s$
- We want: $(1 - a^3)^s \leq g$ (g is acceptable probability of failure)
- $s \geq \ln(g)/\ln(1 - a^3)$
- Since $x < -\ln(1 - x)$ for $0 < x < 1$, we get: $s \geq \ln(1/g)/a^3$
- For $g = 10^{-2}$ and $a = 0.3$, we get $s \approx 180$

Given a set of n ligands $\{M_1, \dots, M_N\}$, Multi-Probe performs the following two steps:

1. Extract invariants from M_1 and M_2 by calling $\text{Pair-Probe}(P_1, P_2)$ on every pair of conformations of the two ligands
2. Test each candidate invariant S obtained at Step 1 against every ligand M_i , $i = 3, \dots, N$ by calling $\text{Pair-Probe}(S, P)$ on S and each conformation P of M_i

2.3 Some experimental results

Figure 4 shows some examples used in the experiments: 1TL, 4TMN, 5TMN, and 6TMN (inhibitors of thermolysin).

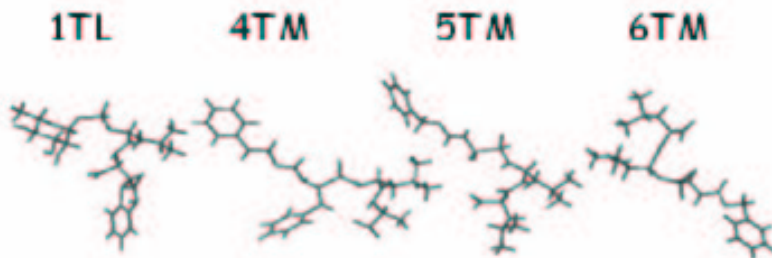


Figure 4: 1TL, 4TMN, 5TMN, and 6TMN (inhibitors of thermolysin)

1TLP has 69 atoms and 10 torsional degrees of freedom, 4TMN has 68 atoms and 15 degrees of freedom, 5TMN has 64 atoms and 13 degrees of freedom, and 6TMN has 63 atoms and 12 degrees of freedom.

In the conformation search step, it produces four sets of representatives of size 850, 1192, 1024, and 955 for 1TLP, 4TMN, 5TMN, and 6TMN, respectively. To find the invariants in these four molecules, these sets of representatives were given input to the search algorithm using Multi-Probe and Pair-Probe. In this experiment, non-hydrogen atom is considered as separate feature or point. Thus, each conformation had approximately 30 features drawn from 6 features classes (the feature classes here being the atom types).

Figure 5 shows the overlapping portions of the molecules which consists of roughly 7 atoms pharmacophore and an

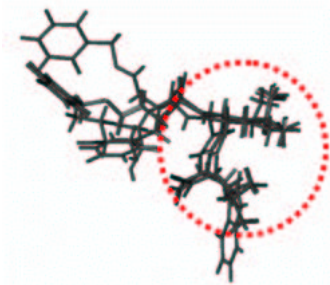


Figure 5: 1TL, 4TMN, 5TMN, and 6TMN overlap in their active conformation

additional 7 atoms of “scaffolding”.

3 The k -nearest neighbors problem

3.1 The problem

One problem to solve in structural bioinformatics is the so called k -nearest neighbors problem : Given a set S of conformation of a protein and a query conformation c , find the k conformations in S most similar to c (with respect to cRMSD, dRMSD or other metric). The solution to this problem is important for biological problems like analysis of MDS and MCS trajectories, clustering conformation to predict structures or graph based methods.

This can be done in time $O(N(\log k + L))$ where :

Figure 6: Proximity along the chain entails spatial proximity

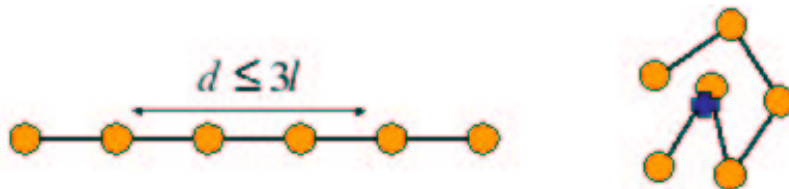


Figure 7: Atoms along the chain are on average spatially distant



- N =size of S
- L =time to compare two conformations

If we want to get the k -nearest-neighbors of all the conformations of S , which is in fact the relevant information in biology, we can compute it in time $O(N^2(\log k + L))$. And this time is much too long for the datasets used in proteomics, which can easily be of 100,000 conformations. Therefore we have to improve the algorithm by some means. There are two ways of dealing with it : Reducing L or building more efficient algorithms (e.g., kD -tree). The idea we will develop here is to reduce L by simplifying the description of the protein.

3.2 The m -average approximation

3.2.1 Idea

Assume that each conformation is described by the coordinates of the n $C\alpha$ atoms. The cRMSD between the two conformations can be computed in $O(n)$ and the dRMSD in $O(n^2)$. But this representation is highly redundant. Proximity along chain entails spatial proximity, 2 $C\alpha$ atoms next to each other on the chain can't go too far away from each other. On the other hand atoms can't bunch up; hence, far away atoms along the chain are on average spatially distant. To get rid of some of this redundancy and speed-up the computation we can do a m -averaged approximation. We cut the backbone into fragments of m $C\alpha$ atoms, then replace each fragment by the centroid of the m $C\alpha$ atoms. This gives us simplified cRMSD and dRMSD speed-up respectively by a factor of $1/m$ and $1/m^2$.

3.2.2 Evaluation: Test Sets

This idea has been implemented and tested [Lotan and Schwarzer, 2003]. It has been tested with 8 diverse proteins (54-76 residues). Each with a decoy set of $N=10,000$ conformations from the Park-Levitt set [Park et al., 1997] and a random set of $N=5,000$ conformations generated by FOLDTRAJ [Feldman and Hogue, 2000]. The test shows us

Figure 8: m -Averaged Approximation

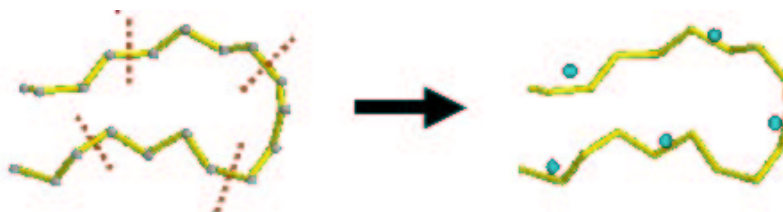
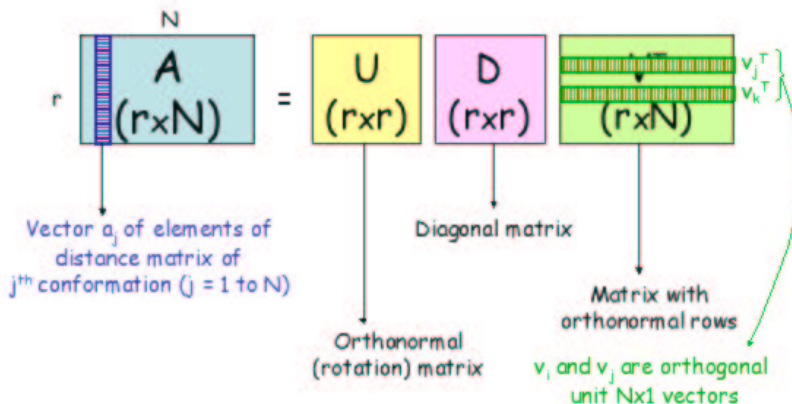


Figure 9: SVD decomposition



that with an m of 9 for the cRMSD and m of 9 for dRMSD we have tolerable loss precision with good speed-up (9x for cRMSD, 36x for dRMSD), and we can gain back the precision in taking a bigger k and then in the set of those k nearest conformation computed with the m -average-approximation, we compute with the exact algorithm the set of k -nearest-neighbors we wanted. There is even a greater correlation for the random set there for greater time saving.

3.3 Further Reduction for dRMSD : SVD decomposition

3.3.1 Idea

The single value decomposition (SVD) is a method used to reduce the dimension of high dimensional data by projecting it on the axes of biggest variance (principal components). We are using during the calculation of dRMSD distance matrices of dimension $n \times n$, $n/m \times n/m$ if we are using m -averaging approximation. those matrices are then compared one to another point per point. This kind of data is suitable for SVD. Thus we will use SVD to speed-up the computation of the k -nearest-neighbors.

3.3.2 Mathematics

We put all our dRMSD distance matrices as vectors in a matrix A the distance matrices being symmetric our matrix A will be of dimension $r \times N$ with $r = 1/2 * n * (n - 1)$ or $r = 1/2 * (n/m) * (n/m - 1)$ if using m -averaging. If a_i is a vector in this matrix, the dRMSD distance between 2 conformations can be calculated by this formula : $dRMSD_m(c_i, c_j) = \sqrt{1/r * ||a_i - a_j||^2}$. Once this matrix build we compute the SVD, which decomposes our matrix A into $A = U D V^T$, where U is an orthonormal (rotation) matrix of dimension $r \times r$, D is a diagonal matrix with the r singular values on its diagonal, and V a matrix of dimension $N \times r$ holding the principal components as vectors. All those vectors v_i are orthonormal to each other. The D matrix is build in such a way that all the singular values are positive and the first singular value (a_1) is the biggest, and all the following singular values are non-increasing in size. So by construction we have in fact our matrix A which gets rotated into a coordinate system defined by U , in which the first axis will be the one with the biggest variability, the second with the second biggest variability, and so on, with the singular values associated so each axis indicating how significant this axis is. The matrix $D * V^T$ the representation of A in this new space. Now by analyzing the singular values choose a value p beyond which the singular values are too small to be significant. We project our points in the new space on the p first principal components. So we reduce the size of our vectors from r to p .

3.3.3 Analysis

For the test set used, we can see that we can reduce the $dRMSD_4^{PC}$ to the summing up of 12 to 20 terms depending on the protein on which we want to do the computation. But we didn't take in account the complexity of the calculation of SVD. The SVD of a $r \times N$ matrix where $N > r$, takes $O(r^2 * N)$ time, and here our $r \sim (n/m)^2$, so the complete time complexity of the SVD is $O(n^4 * N)$, which isn't negligible and only practiced if we use m -averaging.

3.4 Evaluation for 1CTF Decoy Set

In their paper Lotan and Schwarzer (2003) evaluate this speed-up on a data set of 100,000 conformations of the 1CTF protein of the PDB. Wanting to find the 100-nearest-neighbors. If using 4 averaging and 16 principal components they find 70% correct answers with furthest nearest neighbor off by 20%. If using a 6 times bigger k the set obtained contains all the k -nearest-neighbors. To get the 100-nearest-neighbors using the different algorithms they get the following times :

- Brute-force : 84 h
- Brute-force + m-averaging : 4.8 h
- Brute-force + m-averaging + PC : 41 min
- kD-tree + m-averaging + PC : 19 min

So in the end We get a speed-up of more than a factor 200, so the approaches using m-averaging and principal component reduction can be effectively used without information loss.

References

- [1] Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proc. of the Intl. Conf. on Computer Vision*, pages 238–249, 1988.
- [2] A. P. Singh and D. L. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proc. of the Fifth Intl. Conf. on Intelligent Systems for Molecular Biology*, pages 284–293, 1997.