Fast and Efficient Computation of Domain-Domain Interactions from known Protein Structures in the PDB

Panos Dafas¹, Dan Bolser², Jacek Gomoluch¹, Jong Park³, Michael Schroeder¹
¹ Dept. of Computing, City University, London, UK, {panos,msch}@soi.city.ac.uk
² Medical Research Council, Dunn Human Nutrition Unit, Cambridge, UK
³ Biomatics Lab, BioSystems Department, KAIST, Korea

Abstract

Protein interactions provide an important context for the understanding of function. Experimental approaches have been complemented with computational ones, such as PSIMAP ([4]), which computes domain-domain interactions for all multi-domain and multi-chain proteins in the PDB. To determine an interaction, recent PSIMAP algorithms check all residue pairs of any domain pair defined by classification systems such as SCOP ([2]). The computation takes several days for the PDB.

The computation of PSIMAP has two shortcomings: First, PSIMAP considers only interactions of residue pairs rather than atom pairs losing information for detailed analysis of contact patterns. Second, with the superlinear growth of PDB, PSIMAP is not sustainable. In this paper, we address these shortcomings by developing a family of new algorithms for the computation of domain-domain interactions, which is based on the idea of bounding shapes, which are used to prune the search space. The best of the algorithms improves the old PSIMAP algorithm by a factor of 60 on the PDB. Additionally, the algorithms allow for a distributed computation, which we carry out on a farm of 80 Linux PCs. Overall, the new algorithms allows a computation of the interactions at the more detailed atomic level, which would take months with the old PSIMAP algorithm, in 20 minutes. The combination of pruning and distribution make the new algorithm scalable and sustainable even with the superlinear growth in PDB.

1 Introduction

Recently, with the advent of experimental methods to determine protein interaction, there has been increased interest and a number of public databases provide access to the results of these experiments. However, despite the importance of the protein interaction there is a huge gap between the nearly 1M annotated protein sequences and only ca. 50,000 documented interactions. To address this gap, several computational approaches, which provide a large coverage, have been proposed. One such approach is PSIMAP ([4]), which defines interactions between SCOP superfamilies ([2]) by computing domain-domain interactions for all multi-domain proteins with known structure documented in the Protein Databank PDB ([1]). PSIMAP constitutes an interaction between two protein domains of a structure in PDB if the number of residue pairs within a given distance threshold (e.g. 5 Angstroms), is greater or equal to a given number of residue pairs (e.g. 5 pairs). PSIMAP determines this by checking distances of all residue pairs, which is quadratic in the number of residues per domain. This algorithm takes several days on a single machine to process the whole PDB, which contains some 20,000 structures.

The complexity of this computation is the main reason why PSIMAP considers distances between residues rather than between atoms. The abstraction from atoms to residues is made in PSIMAP by representing each residue by two spatial coordinates, the position of the alpha carbon in the protein backbone and the geometric average position of the side chain atoms. When computing the distance between two residues, PSIMAP considers only the shortest of the four possible distances (side chain to side chain, backbone to backbone, etc.). There are two shortcomings in PSIMAP: Approxi-
The old PSIMAP algorithm requires an expensive pairwise comparison of all amino acids of the two domains, the use of bounding shapes improves on this: Using two bounding boxes it is only necessary to pairwise compare the amino acids in the intersection, which is however still quite large in the second figure. This improves, when the domains are rotated using PCA as shown in the fourth figure. The best fit is obtained using the convex hull of the domains as shown in the fifth figure.

Figure 1: Checking interactions of a coiled coil domain pair of C-type lectins from PDB entry 1kwx, an X-Ray structure of the Rat Mannose Protein A Complexed With B-Me-Fuc shown on the left with RasMol. While the old PSIMAP algorithm requires an expensive pairwise comparison of all amino acids of the two domains, the use of bounding shapes improves on this: Using two bounding boxes it is only necessary to pairwise compare the amino acids in the intersection, which is however still quite large in the second figure. This improves, when the domains are rotated using PCA as shown in the fourth figure. The best fit is obtained using the convex hull of the domains as shown in the fifth figure.

mating atom-atom interactions by residue-residue interactions introduces an error as the size of side chains varies greatly from e.g. Glycine to Arginine, whose side chain is ca. 7 Angstrom long. For the latter the averaging leads to a significant error, when considering interactions at 5 Angstrom. However, the approximation is necessary in the old PSIMAP algorithm to reduce the computational overhead, as the algorithm takes already days at the residue level and would therefore take months at the atom level. Yet, despite the approximation PSIMAP is not sustainable due the superlinear growth of PDB.

In this paper, we address these two shortcomings by devising a family of new algorithms, which intelligently prune the search space and thus reduces the number of residue/atom pairs to be checked for the whole of PDB by a factor of 60. Additionally, the algorithms lend themselves to easy distribution over a PC farm. The combination of the new algorithm and usage of a farm of 80 Linux PCs reduces the detailed PSIMAP computation at atomic level, which would take months with the old algorithm, to 20 minutes. This guarantees that the computation with the new algorithms is sustainable even with the superlinear growth of PDB.

The basic idea of our novel algorithms is to prune the search space by applying bounding shapes to the domains. Interacting atoms of two domains can only be found in the intersection of the bounding shapes of the two domains. All atoms outside the intersection can be discarded. The quality of the pruning depends on how well the bounding shape approximates the domain. However, an important trade-off needs to be made: In general, the better the shape’s approximation to the domain, the more complex its computation may be. Overall, only bounding shapes, whose computation takes substantially less than \( O(n^2) \), are useful, where \( n \) is the number of atoms.

A simple bounding shape is the bounding box. It can be computed in \( O(n) \) and leads already to substantial improvements over the old PSIMAP algorithm (see Fig. 3). However, a box can be a very crude approximation, which depends highly on the rotation of the two domains. Consider e.g. long coiled coil structures, which run along a diagonal of the box (see Fig. 1 three left figures). Thus the box contains a lot of “empty” space and is therefore not a good approximation. But this problem can be easily remedied by rotating the domains and aligning the longest axis of the two domains with the boxes x-axis (see forth figure in Fig. 1). This can be achieved with Principal Component Analysis (PCA), which is also linear, i.e. \( O(n) \). As will be shown in Fig. 3, this method leads to an improvement over the basic bounding box. The above two methods, i.e. the basic bounding box and bounding box with PCA can be iteratively applied by computing new bounding shapes for those atoms of the domains that lay in the intersection. Again, the iteration of the method leads to slight improvements.

However, the best improvements are achieved through the convex hull as a bounding shape, which is a much better approximation of the shape of the domains (see fifth figure in Fig. 1). As a result the intersection of the two domains’ convex hulls characterises the domain interaction very well and leads only to a very few false positives, which need to be discarded in the subsequent detailed analysis. Furthermore, the convex hull can be computed in \( O(n \log(n)) \). Overall, the convex hull algorithm requires 60 times less residue pair comparisons than the old PSIMAP algorithm for the PDB and works very well at the atomic level, as most atoms outside the
Bounding Shape Algorithm

Given: Two domains with residue/atom coordinates

1. A bounding shape for each of the two domains is computed.
2. The bounding shape is swelled by the required contact distance threshold.
3. The intersection of the two swelled bounding shapes is computed.
4. All residue (atom) pairs outside the intersection are discarded and for the remaining residues/atoms the number of residue/atom pairs within the distance threshold is computed. If this number exceeds the number threshold the two domains are said to interact.

Figure 2: The Skeleton for the new algorithms.

interaction interface are immediately discarded. Overall, this new algorithm computes all domain-domain interactions at atomic level within 20 minutes, when distributed over 80 Linux machines.

2 A family of new algorithms

Overall, all the algorithms contain the four stages as shown in Fig. 2. Let us examine them in detail.

Bounding Box (BB) The bounding box algorithm (BB) initially constructs a bounding box around each of the two potential interacting protein domains (step 1 in Fig. 2). The bounding box is defined by two coordinates, the bottom left front \( blf \) and top right back \( trb \) of the box. These two coordinates are computed by initialising the box’s minimal and maximal \( x, y, z \) coordinates and then considering each atom’s coordinates possibly updating the box’s \( blf \) and \( trb \) coordinates. Overall, this takes \( O(n) \), where \( n \) is the number of atoms, as all atoms are considered once.

Both domains’ bounding boxes are then expanded (step 2 in Fig. 2) by the required distance threshold in order to include all the atoms (residues) that are within that distance threshold. In step 3 in Fig. 2, the intersection of the two bounding boxes is computed as all the truly interacting atoms (residues) must be in this intersection. For the resulting atoms (residues) the pairwise distances are computed (step 4).

The bounding box algorithm works well if the main axis of the domain is aligned with either the \( x, y \) or \( z \) axes. If this is not the case (see the third figure in Fig. 1), then the search space pruning is not effective and the number of the atom (residue) contacts to be verified at the second stage can be high. Using Principal Component Analysis we tackle this problem of wrong alignment.

Principal Component Analysis (PCA) Often the fit of a bounding box can be improved by an appropriate rotation of the two domains. One such rotation determines the three major axes of the two domains and aligns these major axes with the bounding boxes’ \( x, y, z \)-axes. Such a rotation is performed by Principal Component Analysis (PCA). The computation of PCA is linear in the number of atoms/residues and therefore feasible. After the rotation (see the fourth figure in Fig. 1) the algorithm proceeds applying the bounding box approach as described in the previous section.

Iterative Algorithms (IBB and IPCA) We can further improve both bounding box and PCA algorithms by iterating the steps 1, 2, and 3 of the algorithm in Fig. 2, before proceeding to step 4. Intuitively, this means that we iteratively approximate the interaction interfaces of the two domains with the bounding boxes.

For the basic bounding box case (IBB) for instance once we have computed an intersection of the bounding boxes, we do not yet carry out the detailed comparison of step 4, but instead we remove all amino acids from the domains, which are not in the intersection and apply the algorithm again. This time the bounding box will provide an improved bound, as it considers only the remaining part of the domains. The same approach applies to the PCA algorithm (IPCA).

Convex Hull (CH) A better fit to a domain than a bounding box can be achieved using a convex hull, which is represented by a list of triangles, each being a list of three vertices. The complexity of the convex hull algorithm in three dimensions is \( O(n\log(n)) \), where \( n \) is the number of atoms/residues, using the divide and conquer algorithm ([5]). Thus step 1 in Fig. 2 can be efficiently carried out.

Step 2 and 3 are technically more complicated than for the bounding box, but can both be done in linear time. In step 2, the swelling of the convex hull by the distance threshold \( d \), we perpendicularly shift each triangle by the distance threshold away from the convex hull: i.e. for each triangle \( t \), we compute a vector \( v \) perpendicular to \( t \) and pointing out of the convex hull.
Then we compute $v'$ as the norm of vector $v$ and multiply it by the required distance threshold. Finally, we add $v'$ to each of the vertices of $t$, thus shifting the triangle away from the convex hull and swelling it.

In step 3, we need to compute which atoms/residues are in both domains’ convex hull. To this end, we need a method to determine whether a atom/residue $v$ is inside a convex hull or not. While this is easy for a bounding box, it is more difficult for a convex hull. It can be done by using the signed volumes defined by $v$ and the triangles in the convex hull. Essentially it holds that for any point inside a convex hull each of the signed volumes of the tetrahedrons, defined by $v$ and each of the triangles, is positive ([3]). This can be computed in linear time. With this method we check for every atom/residue in domain $D_1$ whether it is inside the convex hull of domain $D_2$, in which case it belongs to the intersection.

With step 3 settled, the algorithm proceeds with step 4 in Fig. 2 and checks in detail how many atoms/residues pairs in the hull intersection are within the distance threshold.

Since the convex hull can finely capture the geometry of a protein domain the intersection of the domains represents the real interacting interface very well. Hence the total number of actual atom/residue contacts that need to be verified, checking only the atom/residues in the domain intersection, is very small. This makes the convex hull so much better than the other algorithms.

3 Distributing the Algorithms

Before we evaluate the pruning capabilities of the above algorithms, we present the second step towards fast and efficient interaction computation: Distributed computation of PSIMAP. To this end, two problems arise: Can the computation be distributed and if how can the load be best balanced to achieve an overall best performance.

The calculation of PSIMAP with the old and new algorithms is a so-called “embarrassingly parallel” problem: i.e. the computation can be easily partitioned into independent subproblems that do not require any communication and can hence be distributed over a loosely coupled network of computers. Each process is then assigned a task of executing the PSIMAP algorithm for a given set of PDB file entries. Hence each process calculates a part of PSIMAP that corresponds to the domain interactions identified by the set of PDB files assigned to that process. After all the processes finish the calculation the results are collected and merged to produce the overall PSIMAP. This distributed implementation of PSIMAP uses the convex hull algorithm, as it is the fastest.

We have tested the PSIMAP calculation in a distributed environment of Linux workstations. The workstations have a processing power varying between 600-800MHz. Each workstation has assigned a task that corresponds to the calculation of a subset of PSIMAP. Since the processing power of the workstations is the same the assigned tasks must also be equal in terms of computation size. We estimate the computation size of one PDB file by a function of the number of protein domains and size of the protein in terms of number of atoms/residues. To simplify our analysis we further assume that the size of the protein is proportional to the size $s$ of the PDB entry. Furthermore, given $p$, the number of domains of a PDB entry, there are $\binom{p-1}{3}$ potentially interacting domain pairs. Overall, we estimate the processing time for one PDB entry of size $s$ with $p$ domains, as $f(s, p) = s \cdot \frac{\binom{p-1}{3}}{2}$.

Given a number $c$ of computers we split the whole PDB into $m \geq c$ sets which are (almost) equal in terms of estimated execution times. $c$ sets will execute in parallel at any time, while the remaining sets have to wait in a central queue until computers become available. If a small value is chosen for $m$, the load may not be evenly balanced, leading to poor performance. The reason for this is that the estimates are not very accurate, so that the variation of execution times of the sets can be large. To this end, we chose $m >> c$, as it achieves a better load-balancing, and thus a shorter overall computation time. Also, we gave priority to sets with large estimated execution times. This results in overlapping long computations with short ones, leading to a further reduction of the overall computation time. Finally, all results are collected and inserted into a database. Using the above load-balancing strategy, we ran the convex hull algorithm for PSIMAP on a farm of 80 Linux-PC.

4 Results and Conclusions

All of the new algorithms give a dramatic improvement in the number of distance comparisons required (see Fig. 3). Even the simple bounding box approach works well. Furthermore, applying PCA leads to improvements and the iterated application of the bounding box - with or without PCA - further reduces the required comparisons. The convex hull, which is the best approximation to the domain shape, does best and often picks ex-
**Figure 3:** Residue comparisons needed for the algorithms. Number of residue pair comparisons (y-axis) for different distance thresholds (x-axis, in Angstrom). In order from top to bottom graph, OLD represents the exhaustive search, BB the simple bounding box method, IBB an iterated version thereof, PCA the rotated bounding box method, IPCA an iterated version thereof, CH the convex hull method, and AA the real number of interacting residue pairs. Left: All of the bounding shape algorithm improve the exhaustive search OLD by at least a factor 10. Right: CH improves BB by a factor of 6. The iterated versions IBB and IPCA slightly improve the basic methods BB and PCA. Rotation with PCA improves the basic bounding box approach BB.

<table>
<thead>
<tr>
<th>PSIMAP</th>
<th>OLD on 1 PC</th>
<th>CH on 1 PC</th>
<th>CH on 80 PCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residue Atom</td>
<td>Days</td>
<td>Months</td>
<td>CH on 80 PCs</td>
</tr>
<tr>
<td></td>
<td>4.5h</td>
<td>15 min</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20h</td>
<td>20 min</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4:** The table shows how the improved algorithm reduces the computation times. While the OLD algorithm is estimated to take months on the detailed atomic level and hence was not feasible the convex hull algorithm distributed over 80 Linux machines dramatically reduces the computation time to 20 minutes. The reason for the new algorithm taking only marginally longer at the atomic level (20 minutes) compared to the residue level (15 minutes) is due to the larger problem size leading to an overall better load balancing and CPU usage.

The results in Fig. 3 documenting the pruning and in Fig. 4 documenting the joint reduction by pruning and distribution show that our approach successfully overcomes two problems of the original PSIMAP computation: Due to the pruning the computation is feasible at the more detailed and accurate atom level and due to the combined effect of pruning and distribution the domain-domain interaction are scalable and will be able to handle the superlinear growth in PDB.

**References**


