Using convex hulls to extract interaction interfaces from known structures

Panos Dafas1,∗, Dan Bolser2, Jacek Gomoluch1, Jong Park3 and Michael Schroeder1,4

1Department of Computing, City University, London, EC1V 0HB, UK, 2Medical Research Council, Dunn Human Nutrition Unit, Cambridge, CB2 2QH, UK, 3Biomatics Lab, BioSystems Department, DaeJeon, KAIST, Korea and 4Biotec and Department of Computing, TU Dresden, 108 01307 Dresden, Germany

Received on October 12, 2003; accepted on February 3, 2004

ABSTRACT

Motivation: Protein interactions provide an important context for the understanding of function. Experimental approaches have been complemented with computational ones, such as PSIMAP, which computes domain–domain interactions for all multi-domain and multi-chain proteins in the Protein Data Bank (PDB). PSIMAP has been used to determine that superfamilies occurring in many species have many interaction partners, to show examples of convergent evolution through shared interaction partners and to uncover complexes in the interaction map.

To determine an interaction, the original PSIMAP algorithm checks all residue pairs of any domain pair defined by classification systems such as SCOP. The computation takes several days for the PDB. The computation of PSIMAP has two shortcomings: first, the original PSIMAP algorithm considers only interactions of residue pairs rather than atom pairs losing information for detailed analysis of contact patterns. At the atomic level the original algorithm would take months. Second, with the superlinear growth of PDB, PSIMAP is not sustainable.

Results: We address these two shortcomings by developing a family of new algorithms for the computation of domain–domain interactions based on the idea of bounding shapes, which are used to prune the search space. The best of the algorithms improves on the old PSIMAP algorithm by a factor of 60 on the PDB. Additionally, the algorithms allow a distributed computation, which we carry out on a farm of 80 Linux PCs. Overall, the new algorithms reduce the computation at atomic level from months to 20 min. The combination of pruning and distribution makes the new algorithm scalable and sustainable even with the superlinear growth in PDB.

Contact: panos@alumni.doc.ic.ac.uk

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 1 M annotated protein sequences and only ∼50 000 documented interactions. To address this gap, several computational approaches that provide a large coverage have been proposed. One such approach is PSIMAP (Park et al., 2001), which defines interactions between SCOP superfamilies (Murzin et al., 1995) by computing domain–domain interactions for all multi-domain proteins with known structure documented in the Protein Data Bank (PDB; Berman et al., 2000). PSIMAP has been used to investigate evolution of interactions and the formation of complexes. In Bolser et al. (2003), the authors identify highly interactive superfamilies such as the P-loop containing nucleotide triphosphate or immunoglobulin and establish that superfamilies, which are highly interactive, occur in many species and can thus be considered among the oldest (www.biomedcentral.com/1471-2105/4/45). They also identify an example of convergent evolution of the subtilisin-like and the trypsin-like serine protease superfamily, which is indicated in PSIMAP by shared interaction partners of these two superfamilies. Last, but not least, the authors used PSIMAP to show known domains of complexer I and II and their potential interactions with each other.

PSIMAP computes such domain–domain interactions by considering pairs of domains part of a multiple domain protein in PDB. It 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 Å) is greater or equal to a given number of residue pairs (e.g. five 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: approximating 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 ~7 Å long. For the latter, the averaging leads to a significant error when considering interactions at 5 Å. However, the approximation is necessary in the old PSIMAP algorithm to reduce the computational overhead as the algorithm takes days at the residue level itself and would therefore take months at the atom level. Yet, despite the approximation, PSIMAP is not sustainable due to the superlinear growth of PDB.

In this paper, we address these two shortcomings by devising a family of new algorithms that intelligently prune the search space and thus reduce 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 min. 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 (Fig. 4). 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 (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 remedied easily by rotating the domains and aligning the principal axis of the two domains with the box’s $x$-axis (fourth 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 Figure 4, this method leads to an improvement over the basic bounding box. The above two methods, i.e. the basic bounding box and the bounding box with PCA, can be applied iteratively by the computing new bounding shapes for those atoms of the domains that lie 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 (fifth figure in Fig. 1). As a result the intersection of the two domains’ convex hulls characterizes 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 interaction interface are immediately discarded. Overall, this new algorithm computes all domain–domain interactions at atomic level within 20 min, when distributed over 80 Linux machines.

**A FAMILY OF NEW ALGORITHMS**

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

Fig. 2. The Skeleton for the new algorithms.

Bounding box. The bounding box (BB) algorithm initially constructs a BB around each of the two potential interacting protein domains (step 1 in Fig. 2). The BB is defined by two coordinates, the bottom left front, blf, and top right back, trb, of the box. These two coordinates are computed by initializing the box’s minimal and maximal x, y, z coordinates and then considering each atom’s coordinate’s 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’ BBs 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 Figure 2, the intersection of the two BBs 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 BB algorithm works well if the principal axis of the domain is aligned with 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. We tackle this problem of wrong alignment by using PCA.

Principal component analysis. Often the fit of a BB can be improved by an appropriate rotation of the two domains. One such rotation determines the three principal axes of the two domains and aligns these principal axes with the bounding boxes’ x-, y-, z-axes. Such a rotation is performed by 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 BB approach as described in the previous section.

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

For the basic BB case (IBB) for instance, once we have computed an intersection of the BBs, 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 BB 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. A better fit to a domain than a BB 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 (Preparata and Hong, 1977). Thus step 1 in Figure 2 can be efficiently carried out.

For steps 2 and 3, which are technically more complicated than for the BB, (Fig. 3). They 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 $t$ 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 an atom/residue $v$ is inside a convex hull or not. While this is easy for a BB, 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 (O’Rourke, 1998). 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 (third figure in Fig. 3).

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 capture finely 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.

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 so, how can the load be best balanced to achieve an overall best performance?
Computing protein interactions using convex hulls

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 sub-problems 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 and 800 MHz. Each workstation is 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 should 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 the 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 \( \frac{p(p-1)}{2} \) potentially interacting domain pairs. Overall, we estimate the processing time for one PDB entry of size \( s \) with \( p \) domains as \( f(s, p) = s \cdot \frac{p(p-1)}{2} \).

Given a number, \( c \), of computers, we split the whole PDB into \( m \geq c \) sets that 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 choose \( m \gg c \) as it achieves a better load-balancing, and thus a shorter overall computation time. Also, we give priority to sets with large estimated execution times. This results in overlapping long computations with short ones, leading to a further reduction in 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–PCs.

RESULTS AND CONCLUSIONS

All the new algorithms give a dramatic improvement in the number of distance comparisons required (Fig. 4). Even the simple BB 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 exactly the interacting atoms/residues, so that no further checks are needed. Overall, the convex hull approach leads to a 60-fold improvement at residue level. Importantly, the reductions in false positives do not lead to any false negatives.

To further reduce computation time beyond the improvements achieved through the pruning taking place in the new algorithms, we distribute the computation of the convex hull algorithm over 80 Linux–PCs as described above. As summarized in Figure 5, the algorithm computes the interactions for the whole of PDB in 15 min at the residue level and in 20 min at the atom level. This compares with the several days’ computation of the old PSIMAP algorithm at residue level and an estimation of months at the atom level.

The results in Figure 4 documenting the pruning and in Figure 5 documenting the joint reduction by pruning and distribution show that our approach successfully overcomes two
Residue comparisons needed for the algorithms. Number of residue pair comparisons (y-axis) for different distance thresholds (x-axis, in Å). In order, from the top to bottom graph, OLD represents the exhaustive search, BB the simple bounding box method, IBB an iterated version thereof, PCA the rotated BB method IPCA an iterated version thereof, CH the convex hull method and AA the real number of interacting residue pairs. Left: all the bounding shape algorithm improve the exhaustive search, OLD, by at least a factor of 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 BB approach.

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 min. The reason for the new algorithm taking only marginally longer at the atomic level (20 min) compared with the residue level (15 min) is due to the larger problem size leading to an overall better load balancing and CPU usage.

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 interactions are scalable and will be able to handle the super-linear growth in PDB.

REFERENCES


