StructSorter: A Method for Continuously Updating a Comprehensive Protein Structure Alignment Database

Brian Palmer,* Joseph F. Danzer, Kevin Hambly, and Derek A. Debe

Eidogen-Sertanty, Inc., 9381 Judicial Drive, Suite 200, San Diego, California 92121

Received March 20, 2006

Advances in protein crystallography and homology modeling techniques are producing vast amounts of high resolution protein structure data at ever increasing rates. As such, the ability to quickly and easily extract structural similarities is a key tool in discovering important functional relationships. We report on an approach for creating and maintaining a database of pairwise structure alignments for a comprehensive database comprising the PDB and homology models for the human and select pathogen genomes. Our approach consists of a novel, multistage method for determining pairwise structural similarity coupled with an efficient clustering protocol that approximates a full N×N assessment in a fraction of the time. Since biologists are commonly interested in recently released structures, and the homology models built from them, an automatically updating database of structural alignments has great value. Our approach yields a querying system that allows scientists to retrieve databank-wide protein structure similarities as easily as retrieving protein sequence similarities via BLAST or PSI-BLAST. Basic, noncommercial access to the database can be requested at https://tip.eidogen-sertanty.com/.

1. INTRODUCTION

The Structural Genomics Initiative, and structural genomics in general, continues to generate an increasingly large number of experimentally and computationally derived structures. It is well established that the three-dimensional representations of proteins that these methods produce provide key insights into the function of those proteins. One of the challenges of having such a large number of structures available is to be able to efficiently extract a protein of interest and all candidate proteins that have a potentially related function. It has been found that a reliable method for assigning similar function is the topological similarity two proteins share.[1] Moreover, this structural similarity does not need to encompass the entirety of both structures. Since the putative binding sites of a protein largely determine its function, merely aligning two binding regions of the protein is enough to infer similar function. Often, primary sequence similarity analysis will fail to detect functional conservation that can be inferred from structural similarity. Thus, it is increasingly important for biological scientists to have a research tool that allows them to quickly extract structural similarities on a proteome-wide scale.

Many different heuristic methods have been applied to protein structure alignment since the problem is NP-complete.[2] Widely accepted methods include Gerstein & Levitt's,[3] DALI,[4] VAST,[5] SSAP,[6] FAST,[7] and CE[8] (among others[9,10]), which utilize various algorithms (dynamic programming, vector alignment, combinatorial extension, etc.). We have developed a multistage approach that employs several of the key components of the methods listed above. Our approach offers an excellent compromise between speed and reliability. Newer methods such as FATCAT[11] and FlexProt[12] attempt to improve on previous attempts by allowing internal rearrangements of the proteins. For the purpose of performance, our algorithm, like earlier work, treats proteins as rigid bodies. Several of the methods mentioned above, in particular DALI and VAST, are employed to compute frequently updated structural alignment databases. However, these approaches are limited to completing the exhaustive N-by-N comparisons and can require extensive computational resources.

There are several major databases that provide a clustered hierarchy of the Protein Data Bank[13] (PDB). The first, SCOP,[14] is a manually curated classification. Thus, each new entry requires human intervention to determine the structural family in which the protein belongs. The alternatives, CATH[15] and the DALI database, require the N-by-N structural alignments to be precomputed as this is what drives the clustering. Since our clustering protocol drives our structural alignments, we needed a clustering mechanism that was automatic and could be done before any structural alignments were computed. Since high sequence similarity, for the vast majority of cases, leads to high structural similarity, this information was leveraged to determine the clustering. Additionally, common sequence alignment tools such as BLAST[16] are computationally inexpensive, so the clustering adds negligible overhead to our process. By utilizing this protocol to drive our structural alignments, we both dramatically reduce the amount of time to compute all of the necessary structural alignments for the database and ensure that structural alignments will be consistent across similar proteins.

The benefit of having an automatically maintained database of structural alignments is 2-fold. First, as the number of experimental structure submissions continues to grow, the database must be able to easily scale to accommodate the large number of new entries without unnecessary human

* Corresponding author e-mail: bpalmer@eidogen-sertanty.com.

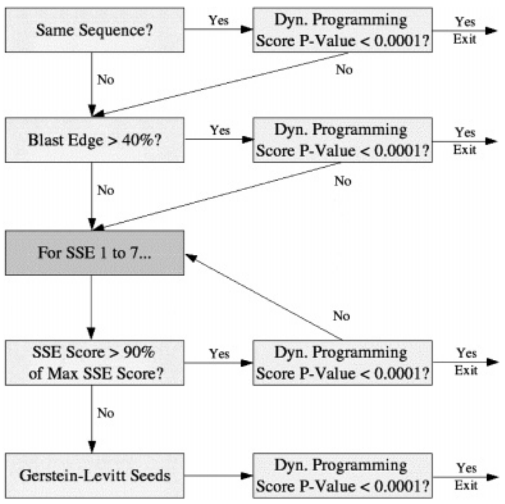
Same Sequence?
Yes
Dyn. Programming Score P-Value < 0.0001?
Yes
Exit
No
No
Blast Edge > 40%?
Yes
Dyn. Programming Score P-Value < 0.0001?
Yes
Exit
No
No
For SSE 1 to 7...
No
SSE Score > 90% of Max SSE Score?
Yes
Dyn. Programming Score P-Value < 0.0001?
Yes
Exit
No
Gerstein-Levitt Seeds
Dyn. Programming Score P-Value < 0.0001?
Yes
Exit

Figure 1. Flowchart of pairwise StructSorter. Various dynamic programming seeding methods are used in order to utilize as much information as is available. We take the first alignment where the dynamic programming step yields an acceptable *p*-value.

intervention. Clearly, requiring exhaustive manpower and/or computer time is not sustainable. Automating as much of the process as possible ensures that updates will occur in a timely fashion. Since users will often be most interested in recently released experimental structures and homology models potentially built from those structures, it is essential that the database be as current as possible. Second, having the precomputed database allows the user to efficiently search for structural homologues that likely have similar function. Having searches complete in a matter of seconds, as opposed to the hours necessary to compute all of the structural similarities, enables a user to try a multitude of permutations in order to extract the most pertinent information and glean new insights into the nature of the data.

2. METHODS

2.1. Pairwise StructSorter. Pairwise StructSorter is a multistage approach utilizing the full range of information available to quickly generate the structural alignment between two protein chains. We use sequence alignment, secondary structure alignment, and previously computed structural alignments to seed an iterative dynamic programming approach to compute structural alignments. The final dynamic programming score is assigned a *p*-value according to its position on an extreme value distribution. Only alignments with a suitable *p*-value are retained. Figure 1 provides a flowchart representation of our hierarchical approach. The flowchart details the various methods we employ for seeding the dynamic programming and the requirements for proceeding from step to step.

In iterative dynamic programming, an initial configuration of two structures is used to provide a starting point for the dynamic programming. We use the same scoring scheme as Gerstein and Levitt.[3] The scoring function is $S(i,j) = M/(1 + (d(i,j)/d)\wedge 2) - \text{shift}$ where $M = 20$, $d = 2.24$, and shift $= 2.5$. This alignment is used to compute the overlay that minimizes the interalpha carbon distances of the paired residues. Dynamic programming is then run on that configuration, and the process is repeated until either the alignments generated for two successive steps are identical, or we have attempted 20 rounds of dynamic programming. A more detailed description is provided in ref 3. The results of iterative dynamic programming vary greatly depending on the quality of the initial alignment used as the seed. As such, we attempt the dynamic programming with seeds generated by various methods. After each attempted seed, if the alignment is deemed satisfactory, we will keep that alignment and quit.

The first seed we try is either the sequence alignment if the percent identity is greater than 40% or the identity alignment if the structures are from the same sequence. This acts as a quick, yet robust solution for all structures with high sequence similarity. The next seed we check is a geometric hashing of the secondary structure to produce an overlay from which to infer an initial structural alignment. Our method is identical to the secondary structure hashing algorithm described by Holm and Sander[17] except for the manner in which we score hits between secondary structure elements (SSEs). While we require the same criteria be met, our score tracks the quality of the match between two SSEs instead of just treating them as good or bad. We base the quality of the match on the angle between the axes of the SSEs and the distance between their midpoints and assess it with the function $(d0 - d)/d0 + (a0 - a)/a0$ where d0 and a0 are the distance and angle cutoffs used by Holm and Sander (4.0 Å and 30.0 degrees, respectively). With this scoring protocol, it is possible to ascertain which secondary structure alignments are actually producing the best overlays. The elimination of false positives allows the algorithm to attempt the best overlays first and avoid generating a "suitable" alignment using a suboptimal overlay. The last seeds we try are those described by Gerstein and Levitt, which are simply aligning beginning, middle, or end of the structures. These trivial seeds merely act as a safety net in case the prior, more intelligent seeding methods fail to produce an acceptable result.

2.2. Significance Assessment. We determine the quality of an alignment by assessing its significance in a manner very similar to that described by Gerstein and Levitt.[18] We first fit an extreme value distribution to the iterative dynamic programming scores of a random sampling of structure pairs. This distribution allows us to attach a statistical significance to the score produced by a given alignment. Since the score is length dependent, we fit a distribution for different length combinations. We found that there were 4511 chains with length less than 70, 13 425 with length between 70 and 140, 19 430 with length between 140 and 250, and 24 536 with length greater than 250. Therefore, we derived parameters for the 10 length combinations: 0-70:0-70, 0-70:70-140, 0-70:140-250, 0-70:>250, 70-140:70-140, 70-140:140-250, 70-140:>250, 140-250:140-250, 140-250:>250, and >250:>250. By employing this scheme, smaller significant alignments will not be overlooked in lieu of larger, inferior alignments.

2.3. Clustering Scheme and Hierarchical Protocol. To completely process a comprehensive database of structures in a timely fashion and ensure alignment consistency, a clustering scheme was utilized. We first clustered all PDB structures with a single-linkage clustering of their representative sequences. This is suitable since very high sequence similarity necessarily leads to high structural similarity. We use the BLAST algorithm to compute all sequence align-

Table 1: Sets of Structures Used during Different Stages of Clustered StructSorter

case	first set	second set
cluster head	all cluster heads	all noncluster heads in clusters where alignment made to head
noncluster head	all cluster heads with alignment to its cluster head	all noncluster heads in clusters where alignment made to head
model	all cluster heads with alignment to its primary template	all noncluster heads in clusters where alignment made to head

ments for the entire database. Then, the criteria for linkage were greater than 90% identity and greater than 95% coverage. Here, we define coverage as the number of residues involved in the sequence alignment divided by the number of residues in the larger sequence. From each cluster, we selected a representative cluster head. First, we chose nonalpha carbon traces over structures with only alpha carbons. Second, X-ray structures were chosen over NMR. The structure with the highest X-ray resolution was chosen. Then, if two structures have equal resolutions, we chose based on the effective length of the structure. We define effective length as the span of the structure minus any residues that are included in this span of the primary sequence yet remain unresolved in the tertiary structure.

When we add a new PDB structure to the database or build a homology model, it is either added to an existing cluster, depending on whether it meets the identity and coverage criteria, or a new cluster is formed. Clusters can also merge if a new structure belongs to more than one. Once any changes are made to a cluster, the head of the cluster is recalculated. This approach avoids the time-consuming calculation of having to recompute the clustering from the beginning every time a new structure is added to the database.

Once the entire set of PDB structures was clustered, we applied the pairwise structure alignment algorithm in a hierarchical fashion. As shown in Table 1, there are three types of cases: cluster heads, noncluster heads, and models. For each case, we first compare the protein to an initial set of cluster heads and then to a second set of proteins derived from the significant matches found in the first set. The second set of proteins comprises the members of all clusters whose head was found to be a significant match from the first set. For cluster heads, the initial set is all other cluster heads. The initial set for noncluster heads is the set of all cluster heads to which its cluster head made a suitable alignment. It is the same for models except that we use the structure used as the model's template instead of a cluster head. For alignments made in the second stage of alignment attempts, we use the alignment with the cluster head to seed the alignment attempt to the noncluster head. If this fails to produce an acceptable result, we default to the pairwise algorithm as defined above. This method of "bootstrapping" alignments ensures consistency of alignments within a PDB cluster and often bypasses the more time-consuming secondary structure alignment process.

3. RESULTS

3.1. Comparison to DALI. To test the accuracy and robustness of StructSorter, for every DALI alignment publicly available, we computed an alignment and compared the results. This test set is biased against our method since this is a set of alignments for which DALI is guaranteed to generate significant alignment. Hence, no cases exist in this set where we would have the chance to succeed where DALI might fail. To compare the two alignments, we computed the dynamic programming score using the Gerstein and Levitt scoring function for each alignment. This is necessary so that we are comparing the alignments by the same metric. We computed a comparison score by subtracting the DALI score from ours and dividing by the larger score. The results are summarized in the histogram presented in Figure 2. The histogram indicates that while the overall correctness of the two methods is comparable, our method tends to significantly outperform DALI on a noticeable number of cases where DALI fails to generate a reasonable alignment.

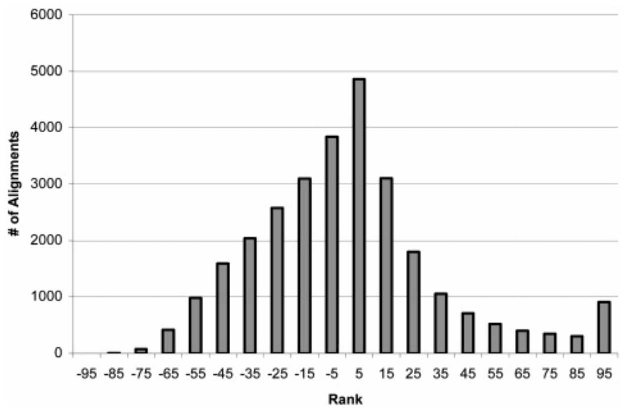

Figure 2. Histogram of StructSorter vs DALI rankings. The rank calculated by subtracting the score of the DALI alignment from that of the StructSorter alignment, dividing by the larger score, and multiplying by 100. Scores are computed using the Gerstein and Levitt scoring function. Positive scores indicate StructSorter found a more accurate alignment.

While the use of the Gerstein and Levitt scoring function may bias the comparison toward our implementation, we have found that it is a better indicator of alignment correctness since it was designed to balance the alignment accuracy, CRMS, with the alignment length. We have found that the DALI scoring function can be biased toward alignment length over alignment CRMS and, as such, can in some cases be a poor indicator of the overall quality of the alignment. For example, Figure 3 displays the overlay based on our alignment and the DALI alignment of PDB 1a02N and PDB 1a3qA. As can be seen, the DALI alignment, which has a DALI score Z-score of 17.6 and an alignment length of 241, in fact has a CRMS of 10.7 and scores poorly with our scoring function. Our alignment scores well with our scoring function with a z-score of 16.9 and has a much lower CRMS of 2.29 and a shorter alignment length of 191.

3.2. Comparison to SCOP. Furthermore, we developed a validation set based on the SCOP classification in order to assess the performance of our significance determination. For every family in the standard classes, we chose two representatives from that set that are also in the 90% PDB

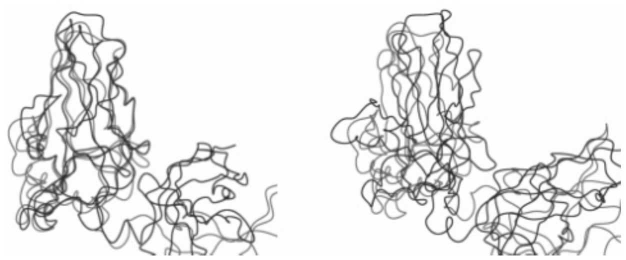

Figure 3. Example of failure of DALI scoring function. The StructSorter result is pictured on the left, while the DALI result is on the right. The StructSorter scoring function scored the alignment on the left well and the alignment on the right poorly, correctly recognizing the poor alignment.

Table 2: Number of SCOP Similarities Found by StructSorter for Various *p*-Value Cutoffs[a]

p−value	family (%)	superfamily (%)	fold (%)	class (%)	none (%)
0.001	96.83	62.76	28.49	0.67	0.08
0.0001	95.52	57.59	21.43	0.03	0.02
0.00001	93.15	42.20	7.20	0.02	0.002

[a] Family level relationships can be confidently assigned as true positives while the class and none level relationships are false positives. Superfamily and fold level relationships are also true positives; however, they will contain a large number of cases with known functional similarity but relatively low geometric similarity. While the difference in the number of class and none similarities kept may appear small, using the 0.0001 cutoff versus the 0.001 cutoff leads to a decrease of roughly 1 100 000 class level false positives and 600 000 none level false positives in the comprehensive database.

select. We ran pairwise StructSorter on each of these structures with every other representative in the same family, superfamily, and fold. Then, we chose a set of 10 structures from each SCOP class to first run against each other to assess the performance of the algorithm at the class level and then run against the other classes to determine how the algorithm behaves when no SCOP similarity is present. The resulting set consists of ∼140 000 pairs.

We chose the *p*-value cutoff of 0.0001 as a balance between retaining true positives yet keeping our database to a reasonable size. As can be seen in Table 2, this cutoff performs best for retaining what we can confidently label as true positives and eliminating the relationships we know are false positives. Limiting the size of our database is important since it has a direct relationship to efficacy of our clustering scheme. As the number of relationships in the database increases, so too does the number of relationships a given cluster head has. Thus, the number of attempts we make at generating alignments will increase dramatically, causing an unacceptable increase in computation time. Moreover, limiting the size of the database is essential for keeping search times to a minimum since there are less data to search against. Overall, we keep the majority of family and superfamily similarities while discarding almost all of the class and no-similarity cases.

It is important to note that the SCOP curators have, for many SCOP similarities in the superfamily and fold levels, utilized additional knowledge such as protein function when assigning proteins to a particular family. The percentage of significant family and superfamily cases we find is consistent with the result obtained by Gerstein and Levitt[17] as well as the recent work by Zhu and Weng where they assessed the performance of their algorithm as well as several others.[8] These results highlight the difficulty in the automated assessment of structural significance, and likely, the incorporation of more information beyond raw geometric similarity will be required to show additional improvement in this area.

3.3. Effect of Clustering Protocol. To ascertain the validity of our clustering protocol, we ran 43 cases against all PDB chains and models in our database. Our database consists of 64 980 PDB chains taken from the recent 21-Jun-2005 release of the PDB in addition to 27 982 homology models built primarily from the human genome. These cases were selected based on chain length and cluster size. Five cluster heads were chosen from large clusters, and five were chosen from small clusters. We chose clusters in this manner to investigate how cluster size affects the number of structural similarities found. Moreover, we chose cluster heads with disparate chain lengths to test how it affects the number found. We also chose a nonredundant set of cases within each head's cluster to gain insight into the effect the bootstrapping was having.

As Table 3 demonstrates, we found that while the size of the cluster caused no appreciable degradation in the number of structural similarities found, the length of the chain did have an impact. This is to be expected when considering large chains are more likely to have a local structural difference. This could cause two chains that have a very significant global structural alignment to have similarity to different shorter chains that would align to only the areas of the larger chains that differ. Of all of the similarities that were found in the unclustered runs that were not found in the clustered runs, none had a *p*-value lower than 1e-7, and the average of all of the *p*-values for these cases was 4.58e-5. Therefore, these were all fairly borderline cases that did not have a very high level of significance.

For the 43 cases used in the clustering analysis, the average time taken to run against the 64 980 PDB chains and 27 982 models in the database was 134 981 s on average. This is roughly 1.45 s per comparison. When run using our clustering scheme, the average time taken was 6846 s. This is nearly a 20-fold reduction in the average time required to compare a structure against the entire database of structures. This allowed us to compute the all-against-all structural similarity database in roughly 1.5 months instead of the 2.5 years required without clustering. The calculations were run on Pentium IV 2.4 GHz processors with 2 GB of RAM.

The most significant benefit of our clustering strategy is in the processing of noncluster heads and model structures. For a recent PDB update to our database comprising 846 cluster heads and 3158 noncluster heads, the average processing time was 8442 and 792 s, respectively. For a cluster that has relatively few neighboring clusters, the vast majority of proteins are true negatives, and our clustering scheme effectively prunes away the need to explicitly compute most of these insignificant alignments.

3.4. Application for Structure−Function. While the link between structural similarity and functional similarity is well established,[1] to specifically demonstrate the value of our database of StructSorter-computed structure alignments for efficient structure-based functional prediction, we performed a search using an example from the literature where structure had been previously used to infer function in the absence of sequence similarity. The protein MJ0577 from *Methanococ-*

Table 3: Results of Analysis of the Effect Clustering Protocol Is Having on StructSorter Efficacy on 43 PDB Cases[a]

case	chain length	cluster size	clustered hits	unclustered hits	% found
1a6m_	150	151	1306	1307	99.92
2bljM	153	151	1304	1304	100.00
1 mlk_	154	151	1304	1304	100.00
2myd_	153	151	1306	1307	99.92
1wvpA	151	151	1302	1303	99.92
1irdA	141	329	1302	1302	100.00
1a0vA	141	329	1302	1302	100.00
1 g9vA	141	329	1302	1302	100.00
1oliA	141	329	1302	1302	100.00
1y22A	141	329	1302	1302	100.00
1jz7A	1011	132	2031	2331	87.13
1bglG	1021	132	2015	2554	78.90
1jz1K	1021	132	2049	2549	80.38
1jz6D	1011	132	2011	2423	83.00
1px4A	1011	132	2038	2354	86.58
1lugA	259	170	235	235	100.00
1a42_	256	170	235	235	100.00
2cbc_	258	170	235	235	100.00
1g48A	258	170	235	235	100.00
1ttmA	258	170	235	235	100.00
1kp8A	525	203	313	314	99.68
1aonA	524	203	312	313	99.68
1j4zA	525	203	312	312	100.00
1mnfA	525	203	313	313	100.00
1svtA	524	203	313	313	100.00
1ccwB	483	6	2053	2337	87.85
1cb7B	483	6	2050	2374	86.35
1i9cB	483	6	2042	2363	86.42
1fx5A	240	6	822	854	96.25
1jxnA	240	6	822	887	92.67
1kblA	872	6	2240	2484	90.18
1dik_	873	6	2243	2772	80.92
1ggoA	873	6	2240	2785	80.43
1jdeA	873	6	2212	2672	82.78
1kc7A	872	6	2238	2515	88.99
1jf3A	147	8	1302	1302	100.00
1hbg_	147	8	1302	1303	99.92
1j16A	147	8	1302	1302	100.00
1vreA	147	8	1293	1293	100.00
1ckaA	57	7	631	644	97.98
1b07A	58	7	625	643	97.20
1m30A	58	7	570	579	98.45
1m3cA	60	7	604	615	98.21

[a] Cluster heads are shown in bold.

cus jannaschii (PDB: 1mjh) was previously discovered to have ATP-binding activity and function as an ATPase only after it was serendipitously found to cocrystallize with ATP during its experimental structure determination.[19] We used PDB 1mjh as the query of our database and within a search time of 5 s returned 53 nonredundant structures from the PDB with superfamily level structural homology to MJ0577 (Z-scores between 5.05 and 14.38). Top scoring results included several ATP-binding proteins, including electron transfer flavoprotein and N-type ATP pyrophosphatase. Using the computed alignments to overlay these structures onto MJ0577 reveals that all of the key H-bonding contacts to ATP are conserved, confirming that MJ0577 could be annotated as a likely ATP-binding protein via structural homology, even if it had not been cocrystallized with ATP. The rapid turnaround time for search, retrieval, and overlay of MJ0577 with its structural homologues demonstrates the utility of our approach for structural homology-based functional inference.

4. DISCUSSION

We have developed a unique approach to computing and maintaining a comprehensive database of structural alignments. Utilizing a clustered database of PDB chains is a key component of this approach. Our clustering protocol, which is both automatic and based on the PDB representative sequences' alignments, drives the construction of our database of alignments, allowing its completion in a reasonable amount of time. We have shown that this clustering does not cause a significant degradation in the coverage of our structural alignments. The automated aspect of our procedure easily accommodates updates to the PDB and the processing of new homology models. This is essential for keeping the database up-to-date with minimal human intervention and maintenance. The clustering protocol is sufficiently flexible such that nearly all pairwise structural alignment algorithms would work within its framework. However, due to the use of disparate scoring functions, the parameters used to assess the significance of the alignment would have to be rederived. Thus, using a variety of structural alignment methods to compute the database would be significantly more complicated and computationally prohibitive.

The iterative dynamic programming approach to solving the structural alignment problem continues to be validated by its role in recently developed algorithms such as FAST.[7] Moreover, our approach is broad and general enough to incorporate additional methodologies for seeding the dynamic programming like the novel approach utilized by FAST. Still, we have found the overall efficacy of our algorithm to be on par with the widely accepted DALI algorithm. We have tuned the significance criterion to the SCOP database and found it to do extraordinarily well at eliminating undesirable cases. The performance of the algorithm is suitable for computing structural alignments for a database of the entire PDB and a large number of homology models.

Having a precomputed database of structural alignments for not only the PDB but also homology models spanning key genomes is of great utility to the field of structural genomics. As the number of experimental protein structures grows, so too will the number of homology models, resulting in an increasingly cumbersome amount of structural data. The ability to search through these data and find proteins with similar structure to a protein of interest is key to assigning function and discovering proteins that share function. Our system enables this ability, and the data available will continue to become richer as the number of experimental protein structures increases.

REFERENCES AND NOTES

(1) Thornton, J.; Todd, A.; Milburn, D.; Borkakoti, N.; Orengo, C. From structure to function: Approaches and limitations. *Nat. Struct. Biol.* **2000**, *7*, Suppl. 991−994.
(2) Lathrop, R. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Eng.* **1994**, *7*, 1059−1068.
(3) Gerstein, M.; Levitt, M. Using Iterative Dynamic Programming to Obtain Accurate Pairwise and Multiple Alignments of Protein Structures. *Proceedings of the Fourth Internation Conference on Intelligent Systems in Molecular Biology;* AAAI Press: Menlo Park, CA, 1996.
(4) Holm, L.; Sander, C. Protein Structure Comparison by Alignment of Distance Matrices. *J. Mol. Biol.* **1993**, *233*, 123−138.
(5) Gibrat, J.; Madej, T.; Bryant, S. Threading a database of protein cores. *Proteins* **1995**, *23*, 356−369.

(6) Taylor, W. Protein structure comparison using iterated double dynamic programming. *Protein Sci.* **1999**, *8*, 654−665.
(7) Zhu, J.; Weng, Z. FAST: A Novel Protein Structure Alignment Algorithm. *Proteins* **2005**, *58*, 618−627.
(8) Shindyalov, I.; Bourne, P. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng.* **1998**, *11*, 739−747.
(9) Lackner, P.; Koppensteiner, W.; Sippl, M.; Domingues, F. ProSup: a refined tool for protein structure alignment. *Protein Eng.* **2000**, *13*, 745−752.
(10) Falicov, A.; Cohen, F. A Surface of Minimum Area Metric for the Structural Comparison of Proteins. *J. Mol. Biol.* **1996**, *258*, 871−892.
(11) Yuzhen, Y.; Godzik, A. FATCAT: a web server for flexible structure comparison and structure similarity searching. *Nucleic Acids Res.* **2004**, *32*, 582−585.
(12) Shatsky, M.; Nussinov, R.; Wolfson, H. Flexible protein alignment and hinge detection. *Proteins* **2002**, *48*, 242−256.
(13) Berman, H.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T.; Weissig, H.; Shindyalov, I.; Bourne, P. The Protein Data Bank. *Nucleic Acids Res.* **2000**, *28*, 235−242.
(14) Murzin, A.; Brenner, S.; Hubbard, T.; Chothia, C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* **1995**, *247*, 536−540.
(15) Orengo, C.; Michie, A.; Jones, S.; Jones, D.; Swindells, M.; Thornton, J. CATH- A Hierarchic Classification of Protein Domain Structures. *Structure* **1995**, *5*, 1093−1108.
(16) Altschul, S.; Gish, W.; Miller, W.; Myers, E.; Lipmanl, D. Basic Local Alignment Search Tool. *J. Mol. Biol.* **1990**, *215*, 403−410.
(17) Holm, L.; Sander, C. 3-D Lookup: Fast Protein Structure Database Searches at 90% Reliability. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*; AAAI Press: Menlo Park, CA, 1995; pp 179−187.
(18) Gerstein, M.; Levitt, M. A unified statistical framework for sequence comparison and structure comparison. *Proc. Natl. Acad. Sci. U.S.A.* **1998**, *95*, 5913−5920.
(19) Zarembinski, T.; Hung, L.; Mueller-Dieckmann, H.; Kim, K.; Yokota, H.; Kim, R.; Kim, S. Structure-based assignment of the biochemical function of a hypothetical protein: A test case of structural genomics. *Proc. Natl. Acad. Sci. U.S.A.* **1998**, *95*, 15189−15193.

CI0601012