doi:10.1016/j.jmb.2006.05.060 J. Mol. Biol. (2006) **361**, 562–590

JMB Available online at www.sciencedirect.com SCIENCE DIRECT® ELSEVIER

Partitioning Protein Structures into Domains: Why Is it so Difficult?

Timothy A. Holland[1], Stella Veretnik[2]*, Ilya N. Shindyalov[2] and Philip E. Bourne[2,3]

[1]*Department of Computer Science, University of California San Diego, 9500 Gilman Dr. La Jolla, CA 92093, USA*

[2]*San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Dr. La Jolla, CA 92093-0537, USA*

[3]*Department of Pharmacology University of California San Diego, 9500 Gilman Dr. La Jolla, CA 92093, USA*

**Corresponding author*

This analysis takes an in-depth look into the difficulties encountered by automatic methods for domain decomposition from three-dimensional structure. The analysis involves a multi-faceted set of criteria including the integrity of secondary structure elements, the tendency toward fragmentation of domains, domain boundary consistency and topology. The strength of the analysis comes from the use of a new comprehensive benchmark dataset, which is based on consensus among experts (CATH, SCOP and AUTHORS of the 3D structures) and covers 30 distinct architectures and 211 distinct topologies as defined by CATH. Furthermore, over 66% of the structures are multi-domain proteins; each domain combination occurring once per dataset. The performance of four automatic domain assignment methods, DomainParser, NCBI, PDP and PUU, is carefully analyzed using this broad spectrum of topology combinations and knowledge of rules and assumptions built into each algorithm. We conclude that it is practically impossible for an automatic method to achieve the level of performance of human experts. However, we propose specific improvements to automatic methods as well as broadening the concept of a structural domain. Such work is prerequisite for establishing improved approaches to domain recognition. (The benchmark dataset is available from http://pdomains.sdsc.edu).

Keywords: structural domains; benchmark dataset; performance evaluation; topological assessment; integrity of secondary structures

Introduction

Partitioning complex protein structures into simpler components is an important feature of proteomic research. The degree of decomposition may vary, but frequently stops at the level of a structural domain. A structural domain is the smallest unit that still possesses characteristics we associate with the whole protein: it is compact, has a hydrophobic core, it can fold and (sometimes) function independently of the rest of the structure. Furthermore, it is the combinatorial assembly of domains that generates the protein universe as we know it today: over two-thirds of prokaryotic and over three-quarters of eukaryotic proteins are composed of two or more structural domains, as predicted from the analysis of completed proteomes.[1] Moreover, domains are basic structural units upon which structural classifications are built and functional assignments performed. It is, therefore, imperative to be able to decompose any 3D structure into domains consistently and reliably. Domain decomposition is performed well by human experts, but is quite difficult for algorithms. The major difficulty lies in the diverse nature of domains, there are always some domains that defy the basic principles and create conflicting situations for the algorithms. Nevertheless, automatic methods are vital in the post-genomic era, since the flood of new structures simply overwhelms the capacity of human experts. Additionally, automatic assignments are uniform and easily reproducible, while assignments by human experts are not necessary so. As we showed previously, human experts cannot agree on domain assignments for approximately 10% of existing structures.[2]

Algorithms for domain decomposition first appeared in 1974;[3–5] these methods used 25 known protein structures in their analysis. Thirty years later, with protein structures in the tens of thousands, new methods for domain decomposition still continue to

E-mail address of the corresponding author: veretnik@sdsc.edu

appear, each trying to reach a bit further in successfully assigning complex structures and identifying domains with difficult architectures. With many methods available today it is imperative to be able to cross-compare their performance, as well as have some understanding of the problems each method encounters.

The results of benchmarking of several automatic methods using a comprehensive dataset of structures were reported recently.[2] However the above dataset was enriched in one-domain structures, reflecting bias similar to PDB. During detailed analysis it became apparent that the majority of problems algorithms and experts encountered were in the multi-domain structures. However, the collection of multi-domain structures annotated by the authors was small and consequently we could not adequately analyze the performance of automatic methods on multi-domain structures. Additionally, biased representation of different architectures (and combination of architectures) in the dataset could have biased the outcome of the benchmarking process. To enable us to perform a sensitive and comprehensive analysis of automatic methods, we constructed a much improved benchmark dataset by painstakingly curating the literature for multi-domain structures. The new benchmark dataset was then employed in the analysis of four automatic methods for domain decomposition: DomainParser, NCBI, PDP and PUU. We investigated successes as well as difficulties each method encountered using a broad spectrum of parameters. Where possible we link the assumptions and rules encoded in each algorithm to the performance of the individual method.

Results and Discussion

Construction of the benchmark dataset

Our goal is to assemble a comprehensive benchmarking dataset in order to evaluate automatic domain assignment methods. Several benchmarking datasets already exist: the 55 chain set compiled by Jones[6] and datasets by Islam (2363 chains).[7] The chief drawback of these earlier benchmarks is the fact that they represent the opinion of a particular expert and thus are likely to have a certain bias. As stated above, experts frequently disagree on domain assignments,[2] particularly for proteins with complex structures. Our approach to constructing a benchmarking dataset is to include proteins for which three expert methods,† CATH,[8] SCOP[9] and AUTHORS,[7] agree on the domain partitioning (in terms of assigned number of domains). We realize that this approach is probably missing the most complex and contentious cases for which there is no unique solution (hence the disagreement among experts). Our intention, however, is to focus on the subset of structures for which a single unambiguous solution of domain partitioning exists, as demonstrated by consensus among experts. Evaluating automated methods with such a benchmark will highlight areas in which automatic methods are lagging behind human experts.

The benchmark dataset developed in this work is referred throughout this paper as Benchmark_2‡ (in contrast to the dataset used by Veretnik *et al.*,[2] which is referred to here as Benchmark_1) and was built in several steps. First, all chains for which the number of assigned domains agreed between CATH and SCOP were selected, resulting in 20,844 chains. Second, these chains were grouped into topology classes, as defined by CATH: each topology group contained chains with identical class, architecture and topology, a total of 211 topology groups. Finally, for each of the topology groups, domain assignments by authors of the crystallographic or NMR structure were retrieved from the literature. Once the domain assignment was found for one of the structures in the topology group, the chain was included in the benchmark if the number of domains assigned by the authors of the structure agreed with that assigned by SCOP and CATH. Only one chain from each representative topology group was included in the benchmark dataset. Since the above topology analysis was performed at the level of domains, the case of one-domain proteins was straightforward. In the case of multi-domain chains, at least one of the domains in the chain had to be a unique representative of the topology class for the chain to be included in the benchmark dataset. In the cases when authors of the structure disagreed in their domain assignment with SCOP and CATH, the topology group was discarded from further consideration. Inspection of additional chains in the same topology group indicated that assignments by authors were similar or identical for all chains in that group, so there was no need to consolidate domain assignments within the group. The above process was repeated for *k*-domain chains in the PDB, with *k* greater than or equal to 2 and less than or equal to 6. One-domain chains were taken , one representative per homology group as defined by CATH, from Benchmark_1. The resulting Benchmark_2 contains 355 chains: 106 one-domain chains, 138 two-domain chains, 55 three-domain chains, eight four-domain chains, five five-domain chains and six-domain chains (Table 1). This is the most comprehensive consensus-based dataset of multi-domain proteins currently available for testing; its strongest points are: (1) unbiased coverage of structural space, each CATH topology group (on the level of domain) or a combinations of topology groups is represented once per dataset (24 combinations of domain architectures are missing from this

† "Expert method" as used here refers to a domain assignment method which includes human expertise during the process of decomposition of the structure into domains. A such CATH qualifies as an expert method.

‡ Benchmark_2 is called Balanced_Domain_Benchmark_2 in the associated resource; http://www.pdomains.sdsc.edu

Table 1. Distribution of chains in the Benchmark_2 *versus* Benchmark_1 datasets

Type of chain	Number of chains in Benchmark_1	Number of chains in Benchmark_2	Overlap between Benchmark_1 and Benchmark_2	Number of chains in Benchmark_3	Coverage of structural space by chains in Benchmark_2 (using CATH topologies)				
					Class 1 Arch: 5 Topol: 227	Class 2 Arch: 19 Topol: 139	Class 3 Arch: 12 Topol: 368	Class 4 Arch: 1 Topol: 86	Total Arch: 37 Topol: 820
1-domain	318 (85%)	106 (33.7%)	106 (100%)	106 (39.1%)	Arch: 2 Topol: 14	Arch: 9 Topol: 26	Arch: 6 Topol: 2	Arch: 1 Topol: 7	Arch: 18 Topol: 70
2-domain	40 (10.7%)	140 (44.4%)	1 (0.07%)	108 (39.9%)	Arch: 5 Topol: 30	Arch: 8 Topol: 2	Arch: 8 Topol: 2	Arch: 1 Topol: 2	Arch: 22 Topol: 79
3-domain	15 (4.0%)	54 (17.1%)	0	45 (16.6%)	Arch: 4 Topol: 16	Arch: 9 Topol: 1	Arch: 7 Topol: 3	none	Arch: 20 Topol: 67
4-domain	1 (0.27%)	8 (2.5%)	0	7 (2.6%)	Arch: 2 Topol: 2	Arch: 3 Topol: 4	Arch: 5 Topol: 15	none	Arch: 10 Topol: 21
5-domain	0	5 (1.6%)	0	5 (1.9%)	none	Arch: 5 Topol: 6	Arch: 4 Topol: 6	none	Arch: 9 Topol: 12
6-domain	0	2 (0.6%)	0	0	Arch: 1 Topol: 1	none	Arch: 3 Topol: 3	none	Arch: 2 Topol: 2

Values in parenthesis indicate the percentage of the total number of chains. Coverage of structural space is represented using CATH nomenclature. For each class 1–4, the number of architectures covered by data in the new benchmark is indicated by A (e.g. A:9 indicates nine architectures), the number of topologies are indicated by T (e.g. T:18 indicates 18 topologies). The total number of architectures and topologies in CATH v.2.5.1 (regardless of domain numbers) is: class:1, A:5, T:227; class:2, A:19, T:139; class:3, A:12, T:368; class:4, A:1, T:86.

set, because no data on domain positions were found in the literature; see Table 7; 37 topologies); (2) the fraction of multi-domain chains in the dataset reflects their true distribution in genomes: two-thirds of the 315 proteins are multi-domain structures. It should be noted that none of the benchmarks described in this paper deal with genetic domains, domains that span more than one chain.

Benchmark_2 is a more comprehensive version of the Benchmark_1 dataset previously reported by us.[2] Benchmark_1 was built using the same principles of consensus among experts; however, domain annotation of the authors of crystallographic structures (AUTHORS) were taken from Islam et al.[7] While this proved to be a fast way of building a consensus dataset, the resulting benchmark was over-saturated in one-domain chains: 85% of all structures in the Benchmark_1 dataset contain single domain (this represents bias inherent in the PDB). Single domain proteins are typically "easier" structures to assign correctly: first, it is impossible to under-cut a single domain chain (under-cut means predicting less domains than expected); second, a simple algorithm which over-predicts one-domain chains will have a 85% success ratio on such a dataset. To some extent this problem can be corrected by the choice of more complex criteria to measure assignment quality, but having an unbiased benchmark dataset is still a better choice. Over-representation of single domain chains plagues all earlier benchmarks as well. Benchmark_2 also samples structural space more evenly than previous benchmarks; thus it avoids an evaluation bias due to over-representation of some topologies and the absence of others. The details of the Benchmark_2 dataset (with respect to Benchmark_1) are summarized in Table 1. A more stringent version of the Benchmark_2 dataset was created by removing 44 chains for which individual expert methods disagree with each other on the exact position of domain boundaries. This new benchmark contains 271 chains and is referred to here as Benchmark_3§. Half of each benchmark dataset is available‖. We have withheld the other half for the purposes of independent benchmarking of domain assignment algorithms, thereby providing a service to the scientific community.

Evaluating automatic methods

Criteria used in the evaluation process

A number of criteria are used to examine the performance of each domain assignment method against the benchmark data (Table 2). The simplest criterion is the number of domains assigned by a given method. Errors in assignments of domains can be further classified as over-cuts (assigning

§ Benchmark_3 is called Balanced_Domain_Benchmark_3 in the associate web resource.

‖ http://www.pdomains.sdsc.edu

Table 2. A brief description of criteria used in the evaluation of domain methods

Criterion	Description	Benchmark_used
Number of domains	Number of domains assigned by a method as compared to expert consensus	Benchmark_2
Over-cutting	Method's rate of over-cutting domains in comparison to expert consensus	Benchmark_2 and Benchmark_3
Under-cutting	Method's rate of under-cutting domains in comparison to expert consensus	Benchmark_2 and Benchmark_3
Fragmentation	Fragmentation of domain by method	Benchmark_2
Multi-domain profile	Method's performance on each of the multi-domain subsets	Benchmark_2
Automatic consensus	The extent of agreement among all automatic methods	Benchmark_2
Boundaries_consistency_90	Agreement of domain boundaries between the method under comparison and the benchmark requires 90% overlap	Benchmark_3
Topological assessment	Evaluation of method's performance with respect to exact position of domains (90% agreement is required) CATH topology nomenclature	Benchmark_2
Integrity of secondary structures	Method's rate of splitting individual secondary structures between domains	Benchmark_3

more domains than the benchmark) or under-cuts (assigned fewer domains than the benchmark). Evaluation using three of the above criteria is performed for the entire Benchmark_2 dataset as well as on individual multi-domain subsets (two-domain subset, three-domain subset, etc.). Fragmentation of domains (cutting the domain into non-contiguous segments) is an important feature to consider; it reflects the balance between compactness of domains (compact domains often have many fragments/segments) and biologically meaningful domains (these tend to have at most a few fragments per domain). The analysis is also performed using a more stringent criterion of boundary consistency which measures the overlap between domain boundaries assigned by the expert methods and that assigned by an automatic method. Boundary consistency 90 requires 90% overlap between the assignment by experts and by a given method. Topological assessment evaluates the method's performance for different structural topologies, as defined by the CATH topology class. Integrity of secondary structures measures the method's tendency to split individual secondary structures between different domains. These criteria are summarized in Table 2.

Four automatic methods are evaluated using the Benchmark_2 and Benchmark_3 datasets described above. The automatic methods are: DomainParser,[10,11] PDP[12], PUU[13] and the method used by NCBI.[14] The choice of the methods simply reflects their availability to the authors of this work. We at-tempted to evaluate as many of the recently pub<?A3B2 show lished automatic methods as possible, but in some cases neither algorithms, nor domain assignments were forthcoming when authors were contacted.

Evaluating automatic methods using the number of domains

Based on the criterion of correct number of assigned domains, PDP appears to be the most accurate method (85% correct) followed by NCBI (83%), DomainParser (77%), and PUU (74%) (Figure 1). The basic tendencies of each algorithm are apparent from Figure 1. For example, PUU tends to over-cut a significant number of structures (38.5% over-cut, 7.3% under-cut), whereas DomainParser tends to under-cut many structures (4.5% over-cut, 18.5% under-cut). Although PDP is more accurate than other automated methods, it has a tendency to over-cut (11.2% over-cut, 3.8% under-cut). NCBI, on the other hand, shows a balance between over-cut and under-cut types of errors (9.9% over-cut, 7.6% under-cut).

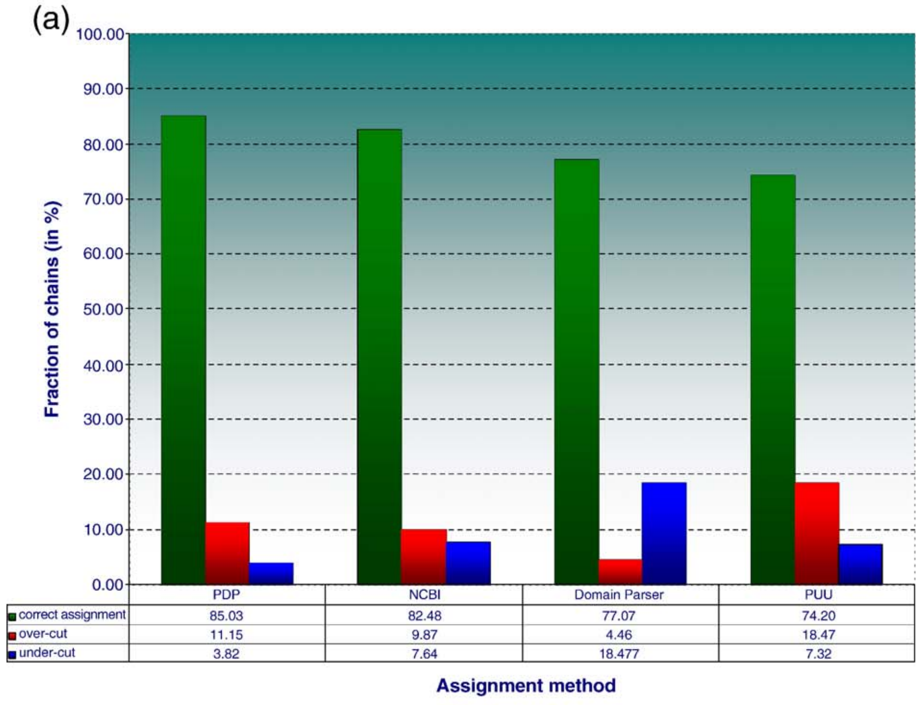
Evaluation by fragmentation of domains

The partitioning of the structure into domains may result in domains consisting of contiguous stretches of polypeptide chain, one stretch per domain (contiguous domains). Frequently, however, regions of the polypeptide chain that are distant in sequence, are close together in 3D structure, thus a domain may consist of two or more segments of the chain, which are non-contiguous in sequence (non-contiguous domains). The different automatic methods vary in their tendency to fragment a domain (Figure 2(a)). We note that the average fragmentation of domains correlates with the average number of domains assigned by each method (Figure 2(b)): thus, if an automatic method assigns on average more domains it also assigns on average more fragments per domain. The ratio between average number of domains and the average number of fragments per domain can be evaluated by comparing the heights of the blue and green bars in Figure 2. In the cases of PUU, PDP and DomainParser this ratio is in favor of average fragments per domain. However, inspection of the expert consensus indicates that the ratio should be inverted: average number of domains exceeding the average number of fragments. The NCBI method is the only automatic method that captures this tendency; however, its average values are higher than that of the expert consensus.

Evaluation using multi-domain profiles

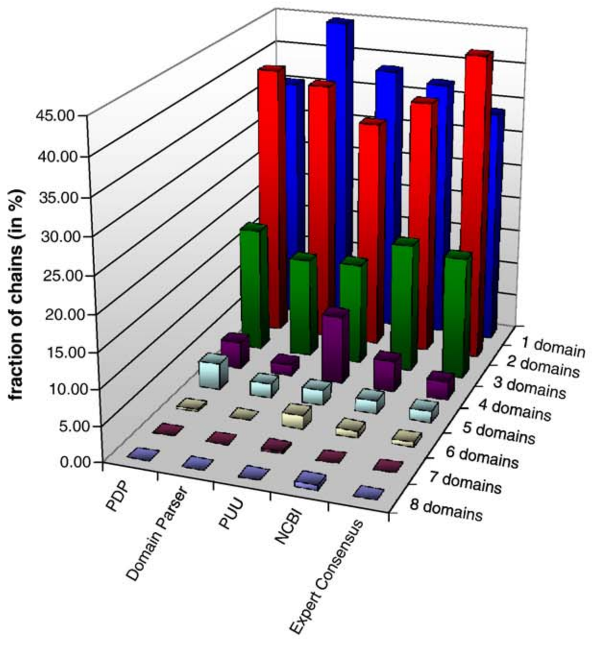
To characterize each automatic method further we subdivided our data into one-domain, two-domain, through six-domain subsets (total of six subsets). We

(a)



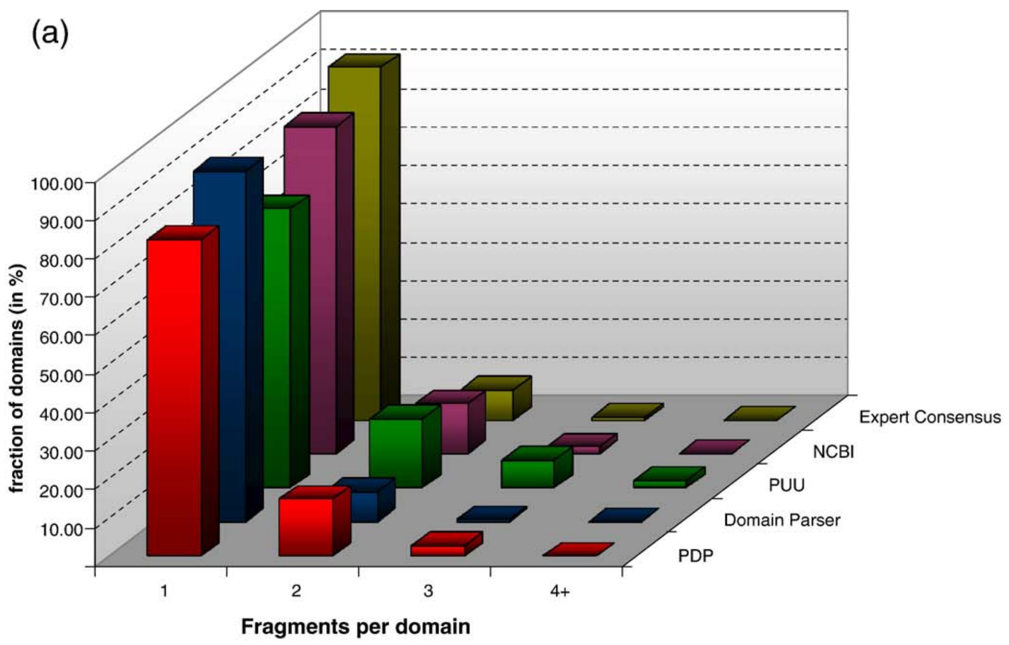
	PDP	NCBI	Domain Parser	PUU
🟩 correct assignment	85.03	82.48	77.07	74.20
🟥 over-cut	11.15	9.87	4.46	18.47
🟦 under-cut	3.82	7.64	18.477	7.32

(b)



	PDP	Domain-Parser	PUU	NCBI method	Expert Consensus
8 domains	0.0	0.0	0.00	0.6	0.00
7 domains	0.0	0.0	0.3	0.00	0.00
6 domains	0.3	0.0	1.9	1.0	0.6
5 domains	3.8	1.9	2.2	1.9	1.6
4 domains	3.8	1.6	9.8	4.4	2.5
3 domains	17.8	14.3	14.6	18.4	17.5
2 domains	39.1	37.5	32.7	36.5	43.8
1 domain	35.2	44.8	38.4	37.1	33.7

Figure 1. Benchmarking of automatic domain assignment methods. (a) Comparison of over- and under-cutting against Benchmark_2. Correct assignment does not take into account location of domains, but only number of domains assigned. (b) Overall number of domains assigned by each automatic method and by expert consensus (in percent).



	PDP	DomainParser	PUU	NCBI method	Expert Consensus
1 fragment	82.4	91.3	72.7	85.0	91.7
2 fragments	14.9	8.0	18.1	13.1	7.6
3 fragments	2.7	0.7	7.2	1.9	0.70
4+ fragments	0.0	0.0	2.0	0.0	0.0

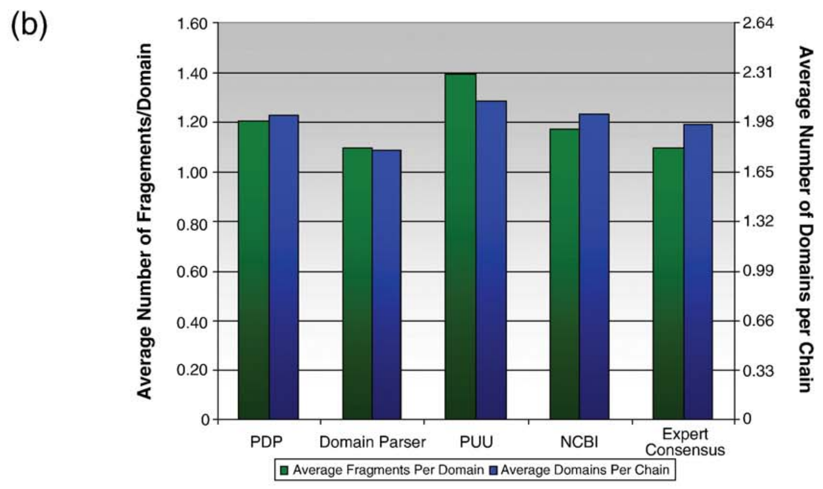
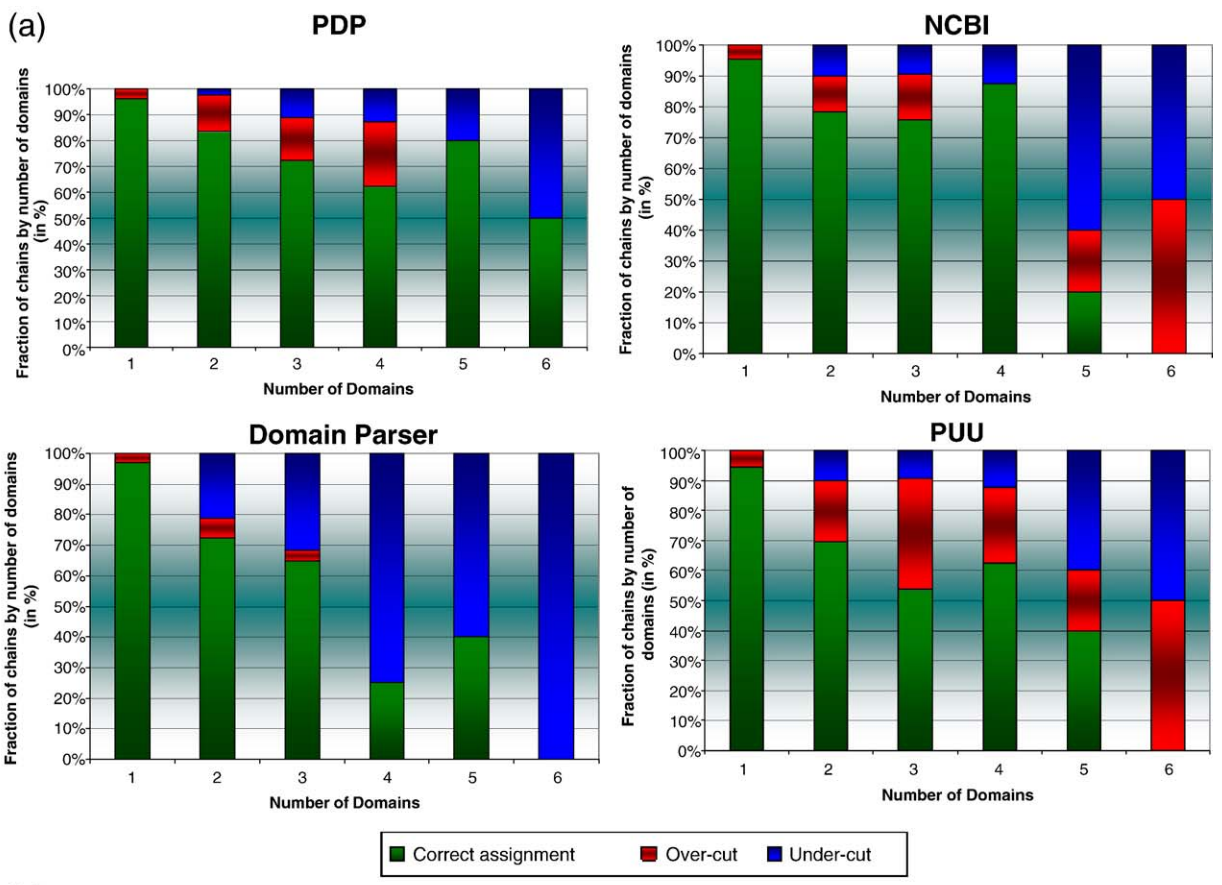


Figure 2. A comparison of domain fragmentation for each automatic domain assignment method. (a) Fragmentation of domains. (b) Side-by-side comparison of the average number of fragments per domain and average number of domains per chain as assigned by each method. The left *Y*-axis scale refers to the average number of fragments per domain and the right *Y*-axis scale refers to the average number of domains per chain. The average number of fragments per domain is calculated using Y/X, where Y is the sum of all fragments assigned for each domain and X is the total number of domains assigned. The average number of domains per chain is calculated using A/B, where A is the sum of all domains assigned for each chain and B is the total number of chains. The proportion between average fragments per domain and average number of domains per chain is 1:1.65.

then tested the performance of each automatic method on each of the subsets (Figure 3).

All automated methods perform at an accuracy of above 90% on single domain chains. This is partly because of the more simplistic nature of single domain chains, but also because under-cutting is impossible on single domain chains, thus leading to more accurate assignment estimates. The accuracy of the methods drops as the number of domains in the structure increases; however, the decrease in accuracy is method-specific, reflecting a distinct approach to domain partitioning used by each method.

(a)



(b)

Domain number		Correct assignment	Over-cut	Under-cut
PDP				
	1	96.3	3.7	0
	2	83.3	14.5	2.2
	3	72.2	16.7	11.1
	4	62.5	25	12.5
	5	80	0	20
	6	50	0	50
DomainParser				
	1	97.2	2.8	0
	2	72.5	6.5	21
	3	64.8	3.7	31.5
	4	25	0	75
	5	40	0	60
	6	0	0	100

Domain number		Correct assignment	Over-cut	Under-cut
PUU				
	1	94.4	5.6	0
	2	69.6	20.3	10.1
	3	53.7	37	9.3
	4	62.5	25	12.5
	5	40	20	40
	6	0	50	50
NCBI				
	1	95.3	4.7	0
	2	78.3	11.6	10.1
	3	75.9	14.8	9.3
	4	87.5	0	12.5
	5	20	20	60
	6	0	50	50

Figure 3. Performance of automatic methods using the multi-domain performance criterion. (a) Each method's performance as measured by the number of assigned domains. For each subgroup containing a specific number of domains, the correct assignment, over and under-cutting rate is shown in green, red, and blue, respectively. (b) A tabular representation of the data plotted in (a).

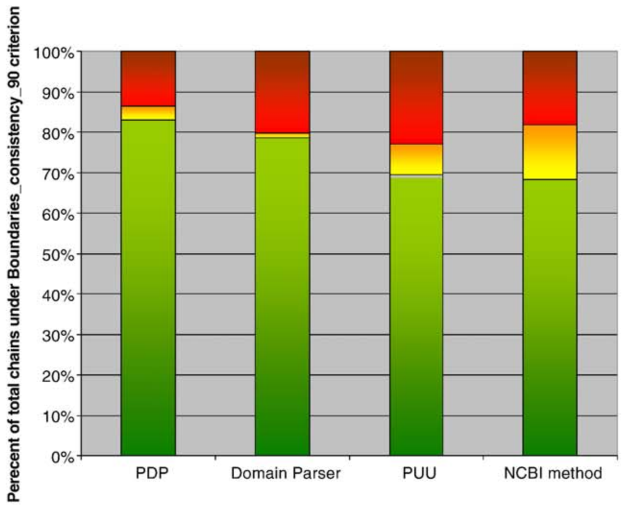
The performance of each method in each data category (Figure 4(a)) can be seen as the method's profile. The most distinct profile is that of DomainParser, where the majority of errors are of the under-cut kind and where the accuracy drops drastically in chains with more than three domains. Proteins of NCBI and PDP show very similar trends on one-domain, two-domain and three-domain chains, but this resemblance breaks down for more complex structures, those containing four or more domains. The performance of PUU is most similar to that of NCBI; however, the fraction of over-cuts is much greater in PUU, which is the most pronounced for two-domain and four-domain chains (when compared to NCBI).

The performance of PDP is consistently superior to other methods; it is particularly impressive on chains with larger number of domains: the method assigns correctly four out of five, five-domain chains and is the only method to correctly assign a six-domain chain. In general the performance of NCBI is very similar overall as well as in its profile character to that of PDP; its assignment of four-domain chains is superior to that of PDP, but NCBI fails to assign correctly most of five-domain chains and both of the six-domain chains.

Additional evaluation by automatic consensus criterion

Another approach to determine to what extent automatic methods agree among themselves on the number of assigned domains is to look at agreement among all automatic methods, i.e. automatic

(a)



	Accurate	Failed domain overlap	Failed domain number
PDP	83	03	14
DomainParser	79	01	20
PUU	69	08	23
NCBI method	68	14	18

(b)

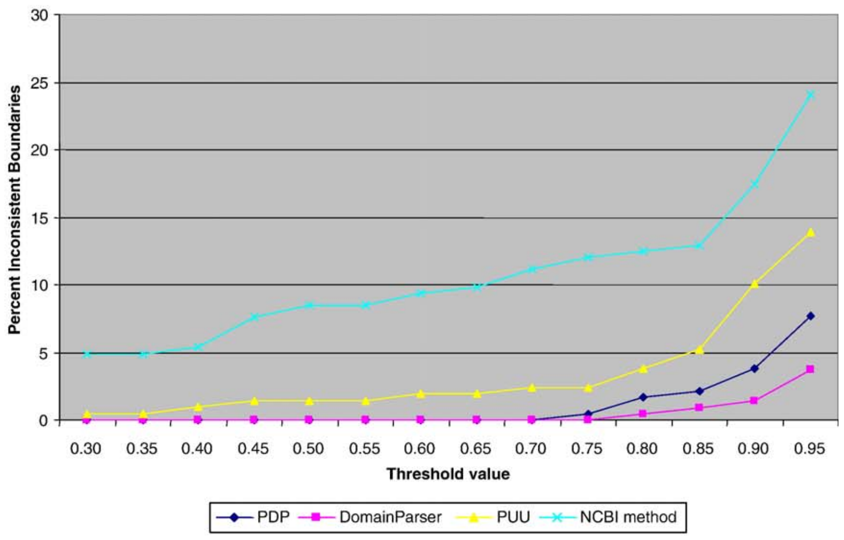


Figure 4. Accuracy of Boundaries_consistency_90 for automatic methods evaluated using Benchmark_3 (a) Overall accuracy of automatic methods at a domain overlap threshold of 90%. Green is accurate assignments, yellow is inaccurate domain boundary, but accurate number of domains, and red is incorrect number of assigned domains (b). Overall performance over an increasing threshold of accuracy.

Table 3. Consensus performance for different numbers of domains per chain

Number of domains per chain	Number of chains reaching consensus using automated methods	Total chains	Percent of chains reaching consensus using automated methods
1	95	107	88.8
2	55	138	39.9
3	17	54	31.5
4	1	8	12.5

consensus (Table 3). The automatic methods agreed completely on 168 out of all 315 chains analyzed, or 53.3% of chains. A large proportion of these chains are single-domain chains, out of 107 single domain chains 95 (88.8%) show complete agreement among automatic methods; however, agreement among automatic methods drops rapidly as the number of domains increases (out of eight four-domain chains only one had complete agreement, while for five-domain or six-domain chains there was no agreement among automatic methods).

Most chains that are partitioned correctly by all methods have clearly identifiable globular domains. Chains that are difficult for all methods are those that consist of several domains that are small and compact, or alternatively represent very large sprawling structures. These situations are usually more difficult for methods because parameters such as connectivity, globularity, preservation of β-sheet structures and other characteristics used by methods are more difficult to capture (see Algorithmic approach to domain decomposition, below).

Evaluating automatic methods using a boundary consistency criterion

Evaluating the performance of a method using only the number of domains it assigns might not be sensitive enough to determine the accuracy of an automated method. At times, methods disagree with experts on exactly where domains start and/or end, i.e. there is disagreement on the placement of domain boundaries, while there is agreement on the number of assigned domains.

To address the issue of overlap between the domains assigned by experts and those assigned by automatic methods we first define a boundary consistency criterion. A boundary consistency of 75% requires that 75% of the domain length, as defined by two domain assignment methods, is the same. Empirically we chose boundary_consistency_90 as a definition of identical assignment by two methods: domain assignments are considered identical if domains overlap for at least 90% of the chain length. Examination of the domain overlap among expert methods revealed that 44 chains in Benchmark_2 did not meet the boundary_consistency_90 criterion, i.e. in these chains different experts positioned domain boundaries differently and the overall overlap between two expert methods was less than 90%. Thus, Benchmark_3 was constructed which excluded these 44 offending chains; this new benchmark consists of 271 chains which have domain numbers and domain boundaries consistently assigned by experts.

Benchmark_3 is used to evaluate the performance of each automatic method by measuring a fraction of chains that meet a given threshold of domain overlap (Figure 4(b)). At each threshold (expressed in percent) the given percent (or more) of the domain must match an expert consensus assignment in order to be considered accurate. All methods with the exception of NCBI perform near perfect up to the threshold of overlap of 75%. At the higher level of overlap, the fraction of incorrectly assigned boundaries increases steadily, yet it does not surpass 15% in the worse case. The domain assignments by NCBI show consistently higher levels of incorrectly placed boundaries: at the lower overlap thresholds the discrepancy is approximately fivefold, while at the highest threshold of 95%, it is roughly twice as large as the remainder of the automatic methods.

Using boundary consenstency_90 criterion we find the relative performance of DomainParser and NCBI changes dramatically. DomainParser, which is less accurate based upon the number of assigned domains, is more accurate in the placement of domain boundaries: thus once the number of domains is determined correctly, the boundaries of domains are nearly always correct. Conversely, NCBI, which performs well using the number of domains criteria, drops noticeably in accuracy when the placement of domain boundaries is considered (Figure 4(b)). That is, for a significant proportion of chains with correctly assigned domain numbers the boundaries of the domains are placed incorrectly. Hence, the order of the performance of the automatic methods changes when we consider boundary consistency, in order of the decreasing performance the order is: PDP (83% correctly assigned domains), DomainParser (79%), PUU (69%), NCBI (68%) (Figure 4(a)).

Topological assessment criterion

The tendency of each method to under-cut and/or over-cut structures is explored further by analyzing

Notes to Table 4:
The first columns represents the CATH topology under evaluation, the second column represents number of occurrences of a given topology in the dataset. Each of the four automatic methods is presented in two adjacent columns. The first column indicates cases of over-cut, the second column cases of under-cut. The number in the cell indicates the number of cases for which miss-assignment (over-cut or under-cut) occurs. A thick frame around a cell indicates that a given miss-assignment is unique to a corresponding method. A red background for a given CATH topology indicates that a given topology is problematic to most/all of the methods. A black background for a given CATH topology indicates that it is easy for most/all methods.

Table 4. Performance of four automatic methods using topological criteria

CATH code	Number of instar	DomPar overcut	DomPar undercut	ncbi overcut	ncbi undercut	PDP overcut	PDP undercut	PUU undercut	PUU overcut
...1_..10_...8	3	–	2	–	–	–	–	–	–
...1_..10_..10	17	–	3	–	2	–	–	–	–
...1_..10_..60	3	–	–	–	2	–	–	–	–
...1_..10_.135	1	–	–	–	–	1	–	1	–
...1_..10_.140	1	–	–	–	–	–	–	–	–
...1_..10_.150	6	–	3	–	1	1	–	–	–
...1_..10_.155	1	–	–	1	–	1	–	–	–
...1_..10_.210	1	–	–	–	–	–	–	–	–
...1_..10_.238	4	–	–	–	–	–	–	1	–
...1_..10_.274	1	–	–	–	–	–	–	–	–
...1_..10_.287	6	–	2	–	3	1	–	–	1
...1_..10_.357	1	–	–	–	–	–	–	–	–
...1_..10_.375	1	–	–	–	–	–	–	–	–
...1_..10_.400	1	–	–	–	–	–	–	–	–
...1_..10_.418	2	–	–	–	–	–	–	–	–
...1_..10_.443	1	–	–	–	–	1	–	–	–
...1_..10_.472	2	–	–	–	–	–	–	–	2
...1_..10_.490	3	1	–	–	–	–	–	1	–
...1_..10_.530	1	–	–	–	–	–	–	–	–
...1_..10_.560	1	–	1	1	–	–	–	1	–
...1_..10_.572	1	–	–	–	–	–	–	1	–
...1_..10_.575	1	–	–	–	–	–	–	1	–
...1_..10_.615	1	–	–	1	–	–	–	1	–
...1_..10_.620	2	–	–	–	–	–	–	–	–
...1_..10_.630	1	–	–	1	–	1	–	–	–
...1_..10_.760	4	–	2	1	2	–	2	–	–
...1_..10_.770	1	–	–	–	–	–	–	1	–
...1_..10_1030	1	–	–	–	–	–	–	–	–
...1_..10_1040	1	–	–	–	–	–	–	–	–
...1_..10_1060	1	–	–	–	–	–	–	–	–
...1_..10_1130	1	–	1	–	–	–	–	1	–
...1_..10_1140	1	–	–	–	–	–	–	–	–
...1_..10_1200	1	–	–	–	–	–	–	–	–
...1_..10_1240	1	–	–	–	–	–	–	–	–
...1_..10_1270	1	–	–	–	–	–	–	–	–
...1_..20_..58	4	–	–	–	2	–	–	–	–
...1_..20_..90	1	–	–	–	–	–	–	–	–
...1_..20_..91	1	–	–	–	–	–	–	–	–
...1_..20_.120	8	–	1	–	–	–	–	1	–
...1_..20_.142	1	–	–	–	–	–	–	–	–
...1_..20_.150	1	–	–	–	–	–	–	–	–
...1_..20_.190	1	–	–	–	–	–	–	1	–
...1_..20_.840	1	–	–	–	–	–	–	–	–
...1_..20_.900	1	–	–	–	–	1	–	–	-
...1_..20_.920	2	–	–	–	–	–	–	–	–
...1_..20_.930	3	–	–	–	–	–	–	2	–
...1_..20_1000	1	–	–	–	–	1	–	1	–
...1_..20_1050	1	–	–	–	–	–	–	–	–
...1_..25_..40	3	2	–	2	–	2	–	2	–
...1_..40_..10	2	–	–	–	–	–	–	2	–
...1_..50_..10	1	–	–	–	–	–	–	1	–
...1_..50_..30	1	–	–	1	–	–	–	1	–
...2_..10_..10	2	–	2	–	2	–	2	–	2
...2_..10_..22	2	–	–	–	–	–	–	–	2
...2_..10_..25	7	–	1	–	2	–	–	–	2
...2_..10_..50	3	–	3	–	3	–	3	–	3
...2_..10_..60	1	–	–	–	–	–	–	–	–
...2_..10_..69	3	–	–	–	–	–	–	–	2
...2_..10_..70	10	–	2	–	7	–	1	–	4
...2_..10_..90	1	–	–	–	–	–	–	–	–
...2_..10_.220	1	–	–	–	–	–	–	–	–
...2_..20_..20	1	–	–	–	–	–	–	–	–
...2_..20_..25	9	–	4	–	–	2	–	–	–
...2_..20_..28	2	–	–	–	–	–	–	–	–
...2_..20_..29	1	–	1	–	1	–	–	–	1
...2_..30_..18	1	–	–	–	1	–	–	–	1
...2_..30_..27	1	–	–	–	–	–	–	–	–
...2_..30_..29	1	–	–	–	–	1	–	–	1
...2_..30_..30	12	–	1	–	3	–	1	–	–
...2_..30_..35	2	–	–	–	–	–	–	–	–
...2_..30_..37	1	–	–	–	–	–	–	–	–
...2_..30_.100	2	–	–	–	–	2	–	–	–
...2_..40_..10	1	–	–	–	–	–	–	–	–
...2_..40_..20	3	–	–	–	1	–	–	–	1
...2_..40_..30	6	–	1	–	1	–	–	–	1
...2_..40_..33	2	–	–	–	–	–	–	1	–
...2_..40_..37	1	–	1	–	–	–	–	–	–
...2_..40_..40	2	–	–	–	–	–	–	–	–
...2_..40_..50	20	–	2	–	1	1	1	1	–
...2_..40_..70	1	–	–	–	–	–	–	–	–
...2_..40_.100	1	–	–	–	–	–	–	–	–
...2_..40_.128	4	–	–	–	–	2	–	2	–
...2_..40_.160	1	–	–	–	–	–	–	1	–
...2_..60_..15	1	–	1	–	1	–	–	–	–
...2_..60_..20	2	–	–	–	–	–	–	–	–
...2_..60_..30	1	–	–	–	–	–	–	–	–
...2_..60_..40	48	3	21	6	4	7	1	12	1
...2_..60_..60	2	–	–	–	–	–	–	1	–
...2_..60_.120	14	–	1	4	–	1	1	2	–
...2_..60_.130	2	–	–	–	–	–	–	1	–
...2_..60_.169	1	1	–	–	–	1	–	1	–
...2_..70_..10	1	–	–	–	–	–	–	–	–
...2_..70_..70	1	–	–	–	–	–	–	–	–
...2_..70_..98	1	–	–	1	–	–	–	1	–
...2_..70_.110	3	–	2	–	–	1	–	–	–
...2_..80_..10	5	–	–	–	3	–	–	–	–
...2_.100_..10	1	–	–	–	–	–	–	1	–
...2_.110_..10	3	3	–	–	–	–	–	–	–
...2_.120_..10	3	–	–	2	–	–	–	–	–
...2_.130_..10	5	3	1	4	–	1	1	1	–
...2_.140_..10	1	–	–	1	–	–	–	–	–
...2_.150_..10	1	–	–	–	–	–	–	–	–
...2_.170_.120	1	–	–	–	–	–	–	–	–
...3_..10_..20	9	–	1	–	1	–	–	–	–
...3_..10_..25	1	1	–	–	–	1	–	–	–
...3_..10_..28	2	–	–	2	–	–	–	–	–
...3_..10_..40	1	–	–	–	–	–	–	–	–
...3_..10_..50	2	–	1	–	–	–	–	–	1
...3_..10_.100	3	–	2	–	–	–	–	–	–
...3_..10_.120	1	–	1	–	–	–	–	1	–
...3_..10_.130	1	–	–	–	–	–	–	–	–
...3_..10_.150	3	–	3	–	–	–	–	3	–
...3_..10_.180	2	2	–	–	–	2	–	2	–
...3_..10_.200	1	–	–	–	–	–	–	1	–
...3_..10_.310	2	–	–	–	–	–	–	–	–
...3_..10_.330	1	–	–	–	–	–	–	–	–
...3_..15_..10	1	–	–	–	–	–	–	–	–
...3_..15_..20	1	–	–	–	–	–	–	–	–
...3_..20_..20	18	–	4	3	–	2	–	4	1
...3_..30_...9	3	–	–	–	–	–	–	–	–
...3_..30_..10	1	–	–	–	–	–	–	–	–
...3_..30_..30	1	–	–	–	–	–	–	–	–
...3_..30_..40	2	–	–	–	–	–	–	–	–
...3_..30_..50	1	–	–	–	–	–	–	–	–
...3_..30_..60	8	–	–	–	–	–	–	–	2
...3_..30_..65	1	–	–	–	–	–	–	–	–
...3_..30_..70	23	–	4	2	2	–	–	5	1
...3_..30_.160	15	–	9	–	5	–	9	–	9
...3_..30_.190	1	–	–	–	–	–	–	–	–
...3_..30_.230	2	–	1	–	1	–	–	–	1
...3_..30_.260	1	–	1	1	–	–	–	1	–
...3_..30_.300	7	–	1	–	–	–	–	–	–
...3_..30_.310	3	–	–	–	–	–	–	–	–
...3_..30_.350	1	–	–	–	–	–	–	–	–
...3_..30_.360	2	1	–	–	–	–	–	–	–
...3_..30_.379	1	–	1	–	–	–	–	–	–
...3_..30_.390	2	–	–	2	–	1	–	1	–
...3_..30_.420	6	–	–	–	–	–	–	–	–
...3_..30_.457	1	–	1	–	–	–	–	–	–
...3_..30_.470	4	–	–	–	–	–	–	–	–
...3_..30_.500	2	–	–	–	–	–	–	–	–
...3_..30_.505	7	–	–	2	–	–	–	1	–
...3_..30_.530	1	–	1	–	–	–	–	–	–
...3_..30_.559	1	–	–	–	–	–	–	–	–
...3_..30_.565	2	–	–	–	–	–	–	–	1
...3_..30_.572	1	–	–	–	–	–	–	–	–
...3_..30_.590	1	–	–	–	–	1	–	1	–
...3_..30_.930	4	–	1	–	1	2	–	1	–
...3_..30_.960	1	–	–	–	–	–	–	–	–
...3_..30_.970	1	–	–	–	–	–	–	–	1
...3_..30_.990	1	–	–	1	–	–	–	1	–
...3_..40_...5	1	–	–	–	–	–	–	–	–
...3_..40_..10	1	–	–	–	–	–	–	–	–
...3_..40_..20	6	–	–	–	–	–	–	–	–
...3_..40_..30	8	–	–	–	–	–	–	–	–
...3_..40_.120	1	–	1	–	–	–	–	–	–
...3_..40_.190	1	–	–	–	–	–	–	–	–
...3_..40_.192	1	–	–	–	–	–	–	–	–
...3_..40_.220	1	–	–	–	–	–	–	–	–
...3_..40_.250	2	–	2	–	–	–	–	–	–
...3_..40_.350	1	–	–	–	–	–	–	–	–
...3_..40_.366	1	–	1	–	–	–	–	1	–
...3_..40_.367	2	–	–	–	–	–	–	–	–
...3_..40_.390	2	1	–	–	–	–	–	–	–
...3_..40_.430	1	–	–	–	–	–	–	–	–
...3_..40_.460	1	–	1	–	–	–	–	–	–
...3_..40_.525	1	–	1	–	–	–	–	–	–
...3_..40_.600	1	–	–	1	–	–	–	1	–
...3_..40_.630	3	–	–	–	–	–	–	1	–
...3_..40_.710	2	–	–	–	–	1	–	1	–
...3_..40_1020	1	–	–	–	–	–	–	–	–
...3_..40_1050	2	–	–	–	–	–	–	–	–
...3_..40_1190	1	–	–	–	–	–	–	–	–
...3_..50_...4	1	–	–	–	–	–	–	–	–
...3_..50_...7	1	–	1	1	–	–	–	1	–
...3_..50_..12	3	–	–	–	–	2	–	–	–
...3_..50_..30	2	–	–	–	–	–	–	1	–
...3_..50_..50	10	–	1	3	–	2	–	3	–
...3_..60_..10	1	–	–	–	–	–	–	–	–
...3_..60_..15	1	–	–	1	–	–	–	–	–
...3_..60_..20	1	–	–	–	–	–	–	–	–
...3_..60_..21	2	–	1	–	–	–	–	–	–
...3_..65_..10	1	1	–	–	–	–	–	–	–
...3_..80_..10	1	1	–	1	–	1	–	1	–
...3_..80_..20	2	–	–	–	–	–	–	–	–
...3_..90_..10	1	–	–	–	–	–	–	–	–
...3_..90_.110	1	–	–	–	–	–	–	–	–
...3_..90_.175	1	–	–	–	–	–	–	1	–
...3_..90_.180	1	–	–	–	–	–	–	–	–
...3_..90_.190	5	–	–	3	–	–	–	2	–
...3_..90_.209	1	–	–	–	–	–	–	–	–
...3_..90_.228	1	–	–	–	–	–	–	–	–
...3_..90_.230	3	–	1	–	–	–	–	–	–
...3_..90_.260	1	–	1	–	–	1	–	1	–
...3_..90_.420	1	–	1	–	–	–	–	1	–
...3_..90_.450	1	–	–	–	–	–	–	–	–
...3_..90_.650	1	–	–	–	–	–	–	1	–
...3_..90_.660	1	–	–	–	–	–	–	–	–
...3_..90_.700	2	–	1	–	–	–	–	1	–
...3_..90_.770	1	–	–	1	–	–	–	1	–
...3_..90_.780	1	–	–	–	–	–	–	–	–
...4_..10_..40	1	–	–	–	–	–	–	–	–
...4_..10_..70	1	–	–	–	–	–	–	–	–
...4_..10_.110	2	–	–	–	–	–	–	–	–
...4_..10_.240	1	–	–	–	–	–	–	–	–
...4_..10_.310	1	–	–	–	–	–	–	–	–
...4_..10_.320	1	–	–	–	–	–	–	–	–
...4_..10_.400	2	–	–	–	2	–	2	–	–
...4_..10_.410	1	–	–	–	–	–	–	–	–
...4_..10_.490	1	–	–	–	–	–	–	–	–
...4_..10_.740	1	–	–	–	–	–	–	–	–
.All_...._....	chains	–	–	–	–	–	–	–	–

the performance of each method on different types of architectures and topologies. Table 4 systematically presents all topologies from Benchmark_2 using CATH nomenclature for class, architecture and topology of the domains. For each topology the total number of occurrences in Benchmark_2 is calculated. Some topologies occur multiple times within the benchmark as they combine with many other topologies to form multi-domain structures (a rudimentary power-law distribution of domains can be observed in this dataset). For each topology the number of times an over-cut or under-cut occurs for a specific method is calculated. For each domain assignment method we provide two adjacent columns, one for over-cuts, one for under-cuts, to get a so-called "topological fingerprint" of each method. Remarkably, fingerprints are specific to each method, reflecting method-specific strengths and weaknesses in partitioning structures consisting of different topologies. Examples of topology-specific behavior follow¶.

(1) PUU which usually assigns more domains than the experts, has a set of specific topologies for which it assigns fewer domains than experts, most of them are confined to several architectures, all in the beta class, all small: ribbons, single sheets, some rolls and a few small barrels (2.10–ribbon; 2.20 single sheet; 2.30–roll; 2.40–barrel).
(2) NCBI exhibits a peculiar characteristic: it errs where all other methods do not: both by over-cutting (2.120.10, 3.10.28) and under-cutting (1.10.60; 1.20.58; 2.80.10; 2.10.70, 2.80.10).
(3) Occasionally PDP is the only method that assigns domains correctly for a given structure, while all other methods err (1.10.560, 2.10.25; 2.20.29; 2.40.30, 3.30.70; 3.30.230).
(4) DomainParser, which tends to assign fewer domains that experts, errs frequently on a specific topology immunoglobulin-like sandwich (2.60.40): out of 48 occurrences of this domain, DomainParser erred in 21 cases (the number of the structures involved is less than 21, most of the structures in this category have multiple copies of the immunoglobulin-sandwich topology).

It should be noted that Table 4 considers an entire structure and not individual domains/topologies, thus errors assigned to some topologies are guilt-by-association. In the case of multi-domain proteins the particular topology (domain) might not be over-cut, but is marked as such because it co-exists in the structure with another topology (domain) that is over-cut. Similarly for undercutting; in some cases of three, four, five and six-domain proteins certain domains are not directly contributing to the error in assignment, even though they appear so in Table 4. We notice that an individual architecture/topology by itself frequently does not determine completely whether structures will be assigned correctly over-cut or under-cut. While certain architectures have a greater tendency to be assigned incorrectly, it is the combination of certain architectures that defines the level of difficulty. This should be evident from inspecting the numbers associated with different topologies in Table 4. Frequently the total number of occurrences of a given topology in the dataset is larger than the number of errors for the same topology.

Analysis of Table 4 provides an overview of method's tendency with respect to topology of the structure. The immediate observation is that all-β class architectures as well as α/β architectures, which are predominantly β-structures, are difficult for all methods. Overall there is a great tendency to under-cut structures that contain all-β domain (s). This phenomenon might involve two or more all-β domains or β-domain in combination with α/β domain(s). In all cases extensive interactions between β-strands confuses all or the majority of methods. Cases of over-cut within all-β class are confined to very large β-structures (Figure 6(d) and (f); Figure 7(g) and (h)). Many cases of over-cut involve guilt-by-association. The over-cut occurs in the other domain of the protein, usually of all-α or α/β class. Not all beta architectures are tricky, most of the rolls and small barrels are assigned correctly.

All-α topologies are easier for all methods to resolve correctly; the general trend among methods is to over-cut. This is particularly evident in the methods which require compactness, specifically PUU and PDP (see below), since all-α architectures may be more protruding and less globular (Figure 7(e)).

Architectures and topologies of the α/β class cause the least problems for all methods. We hypothesize that a combination of alpha and beta secondary structures results in an overall structure with more consistent contact density facilitating correct domain partitioning. There are some difficult α/β architectures, which confuse automatic methods; these usually contain large and convoluted shapes such as three-layer sandwiches and complex architectures.

These problems exist at the level of topology rather than architecture. Nearly every architecture has some "easy" and some "difficult" topologies for automatic methods. Thus, it is not so much the shape of a domain and the orientation of the secondary structure elements within a domain (architecture) which causes difficulty, but rather the connection between secondary structure elements (topology). Intuitively, different connectivity will effect exact positioning between secondary structure elements and motifs (combination of a few secondary structure elements) within the domain. Perturbations within an architecture effect the distribution of contact densities and potentially how the structure is partitioned.

¶ All architectures and topologies are specified using CATH nomenclature.

Integrity of secondary structures

The four automatic methods vary in their approach to splitting a secondary structure between domains. We measured each automated method's tendency to split α-helices and β-strands between domains (Table 5). We also looked into the same property in expert methods CATH and SCOP (AUTHORS does not lend itself easily to this analysis because it is not present in the PDB or another computer parseable form). An α-helix is considered to be split between two domains if two or more of its residues reside in a different domain from the rest. A β-strand is considered to be split if 30% of its residues reside in a different domain from the rest. The fraction of the cut secondary structure elements vary widely among methods. Note that PDP, the best performing method, has the highest fraction of cut secondary structures. NCBI, on the other hand, never cuts a secondary structure and its performance is close to that of PDP. Both expert methods do occasionally cut secondary structure elements.

Summary of performance of each automatic method

Performance of each automatic domain assignment method is evaluated using some of the criteria described above (Table 6). Each criterion can assume the values of worst, best or middle. Among four methods two will assume the middle value, one best and one worst. This approach provides an overall assessment of the strength and weakness of each method; however, it does not quantify differences nor permits direct comparison between two, middle, methods.

Automatic methods: relationship between a method's performance and algorithm

This section examines the collective and individual strengths and weaknesses of the four automatic methods. Where possible we connect the performance issue to specific features of the algorithm behind the method. Finally, we suggest possible solutions to the problems we observe.

Basic principles implemented in each algorithmic approach

DomainParser. DomainParser uses a top-down[a] approach to domain decomposition implemented using a graph theoretical approach. Each residue is modeled as a node and each connection between residues as an edge. A connection between two residues exists either when a residue lies next to another in sequence or is in close proximity in 3D structure. The spatial proximity between two residues requires that the distance between at least one atom from each residue is 4Å or less. The strength of connection between two nodes (referred to as capacity) is proportional to the strength of the interaction between two residues represented by the nodes.

The division of the protein into domains is done by systematically splitting a structure into two parts, which is equivalent to separating a network into two parts using a minimum cut approach. The process of division is then repeated with each individual domain until one of the stop criteria is reached. This constitutes the first step of the algorithm. In the second step (a post-processing step) a number of parameters are used to evaluate the suitability of potential domains generated in step 1. The parameters include compactness, radius of gyration, number of non-contiguous segments per domain and the distribution of domain sizes. The minimum length of a domain is 35 residues and β-strands are not cut unless they act as a narrow polypeptide segment connecting two or more domains.

PUU. PUU is a recursive top-down approach, which uses a hypothetical model of autonomously folding units corresponding to protein domains. The domains are defined as groups of residues that oscillate as a single unit relative to other residue groups. The amplitude of oscillation is determined by oscillating time τ. A greater τ indicates a greater mobility and a greater probability for a given group of residues to reside in separate domains. The oscillation time is proportional in the center of mass of the group and is inversely proportional to the so-called interference strength. The interference strength is calculated by evaluating pair-wise residue–residue interactions, which in turn are calculated for any two atoms in two residues which are 4Å apart or less. A hierarchical five-level filtering process is applied during partitioning of the structure. The process does not require all five conditions to be met. Rather, if the condition for applying the filter is true the lower filters are not tested. The filters applied are: (1) the domain should have at least 80 residues to be considered for cutting; (2) highly flexible units are always cut (defined in terms of τ); (3) β-sheets forming a network are never cut; (4) the cut is accepted if both sub-domains are compact (globularity >0.8); and (5) a cut that yields a small non-globular unit (<40 residues) is accepted on the condition that the situation is rectified in the recursive application of the filters.

NCBI. There is no publication dedicated to the method itself, rather it is used as a step in a process of defining protein cores.[14] The available information on the method is rather sparse. NCBI uses a similar approach to that of PUU. The partition of the structure into two domains is satisfactory when

[a] Top-down approach implies that the method begins with the entire protein and recursively partitions it into small units (domains). Alternatively, a bottom-up approach assembles structural domains from smaller units such as individual secondary structure elements. All four methods discussed here use a top-down approach.

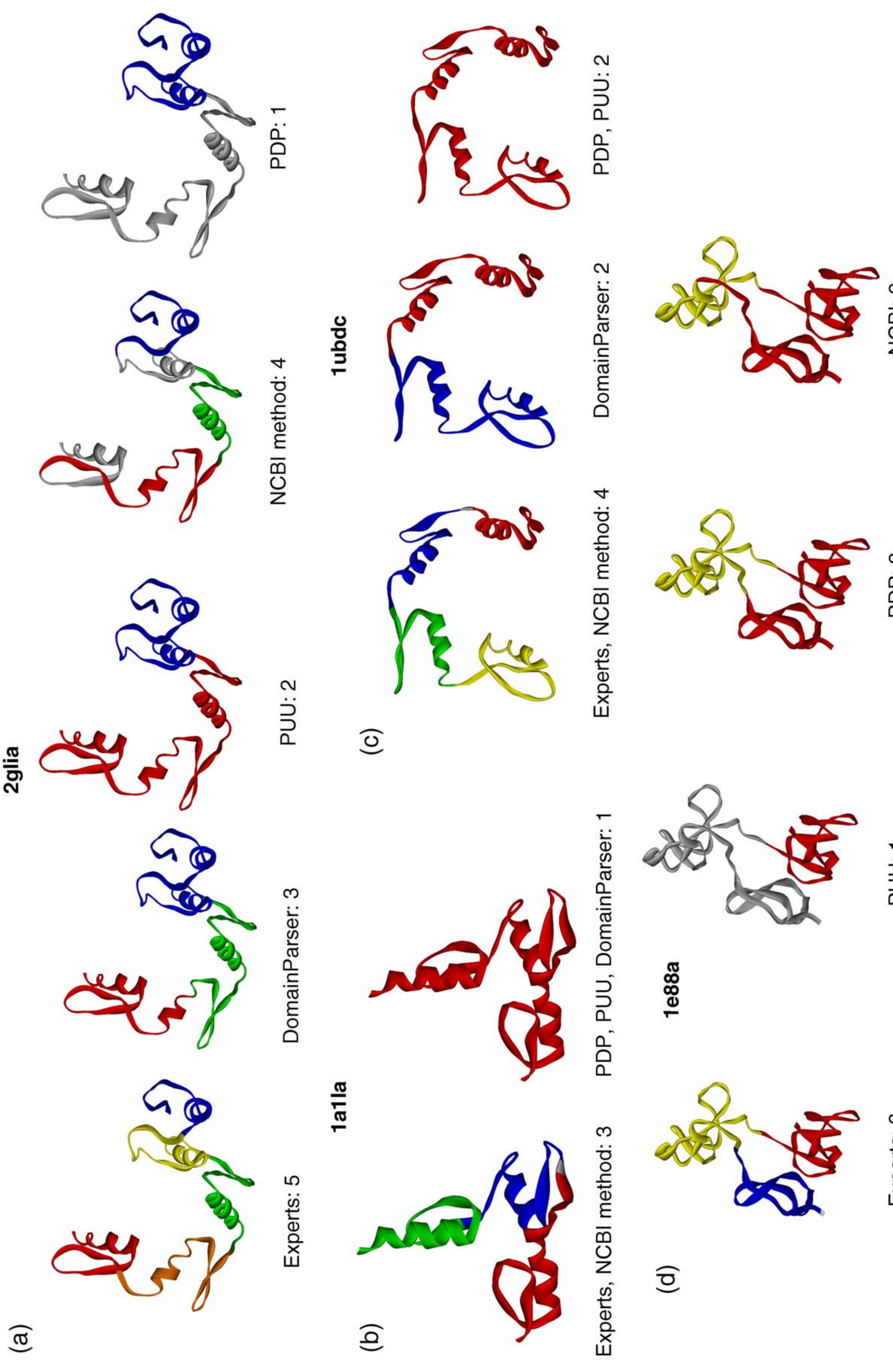


Figure 5 *(legend on next page)*

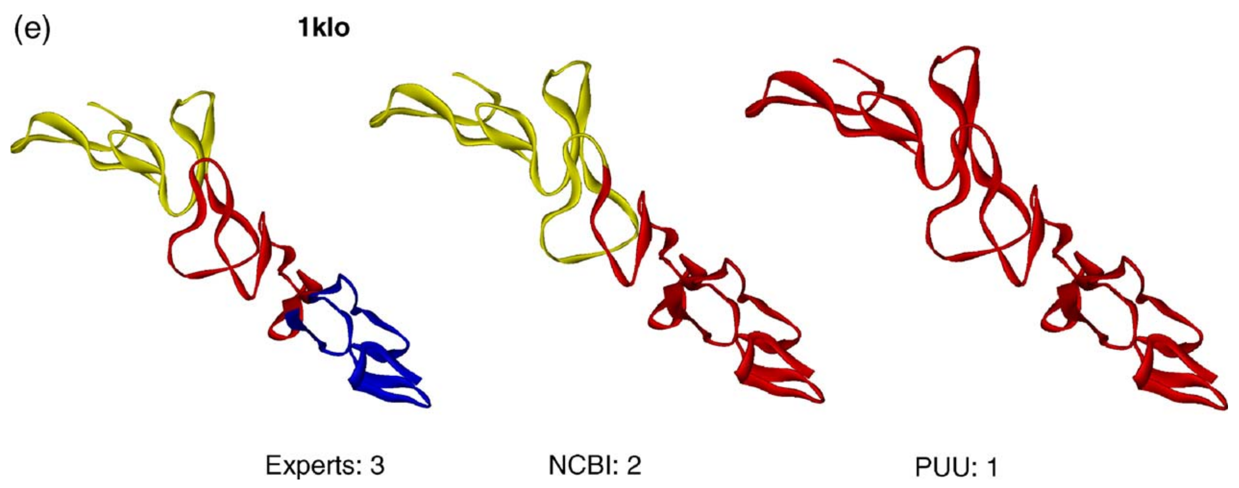


Figure 5. Examples of problematic small domains. Gray coloring represents amino acids not assigned to a domain by the given method.

intra-domain contact density is at least twice as high as the inter-domain contact density. In addition the domains are not permitted to have isolated secondary structure elements, those forming no contacts with the rest of the domain. The domain boundaries are not allowed to go through a secondary structure element, but should be placed in the loops between the elements. The smallest domain must contain at least 25 residues within secondary structures.

PDP. PDP is a recursive algorithm that makes either a single cut producing two contiguous domains or a double cut, where the cuts are at least 35 residues apart and result in spatially close domains. The best cut is selected using criteria of minimum contacts between resulting domains. The extent of the domain–domain contacts is calculated assuming spherical domains: their surface area is proportional to $n2/3$, where n is the number of residues. Additionally the above contact value is normalized by the size of the domains. The threshold on normalized domain contact is set to be 0.5 of the average contact density of the entire domain. The algorithm continues recursively by partitioning each of the resulting domains further until stopping criteria are met. During the post-processing step, the number of contacts between resulting domains is evaluated and domains with a high level of contacts are merged together. Very small domains (below 35 residues) are discarded.

Performance of methods

Observation. All methods have troubles identifying small domains: a small domain frequently gets combined with another small domain or becomes a part of another larger domain.

Threshold on domain size issue. One reason for difficulties in identifying small domains such as Zn-fingers is rather simple. The minimum domain size is set to be around 35–40 residues for PDP, DomainParser and PUU, whereas Zn-finger domain can be as small as 30 residues (Figure 5(a)). We think that minimum domain size is set optimally and cannot be reduced any further in the above methods without producing many incorrect small domains, as the domain size often serves as a stopping criterion to prevent excessive splitting of the structure. NCBI defines minimum domain size somewhat more flexibly. It requires that at least 25 residues should be in the secondary structures, thus very small compact domains with a majority of the residues in α-helices and β-sheets are still acceptable, as in the cases of 1a1la (Figure 5(b)) and 1ubdc (Figure 5(c)).

Possible remedy. For top-down algorithmic approaches the best solution might be “remembering” the architectures of very small but true domains (such as Zn-finger or EGF domains) and checking for the presence of such architectures during the process of subdivision into domains.

Contact density issue. Another type of problem exists with small ribbon-type domains from the all-β class, such as 1e88 (Figure 5(d)) and 1klo (Figure 5 (e)). In these cases the domains pass the size threshold, but all methods have difficulty separating domains due to insufficient contrast in contact density within domains and between domains. On one hand, small domains have lower contact density within the domain because small ribbon domains are very loosely packed. On the other hand, the interface between β-domains is more involved because β-strands tend to contribute to the contacts in the domain interface.

Possible remedy. The critical ratio between inter and intra-domain densities (which is being used as a criterion for splitting/assembling domains) may

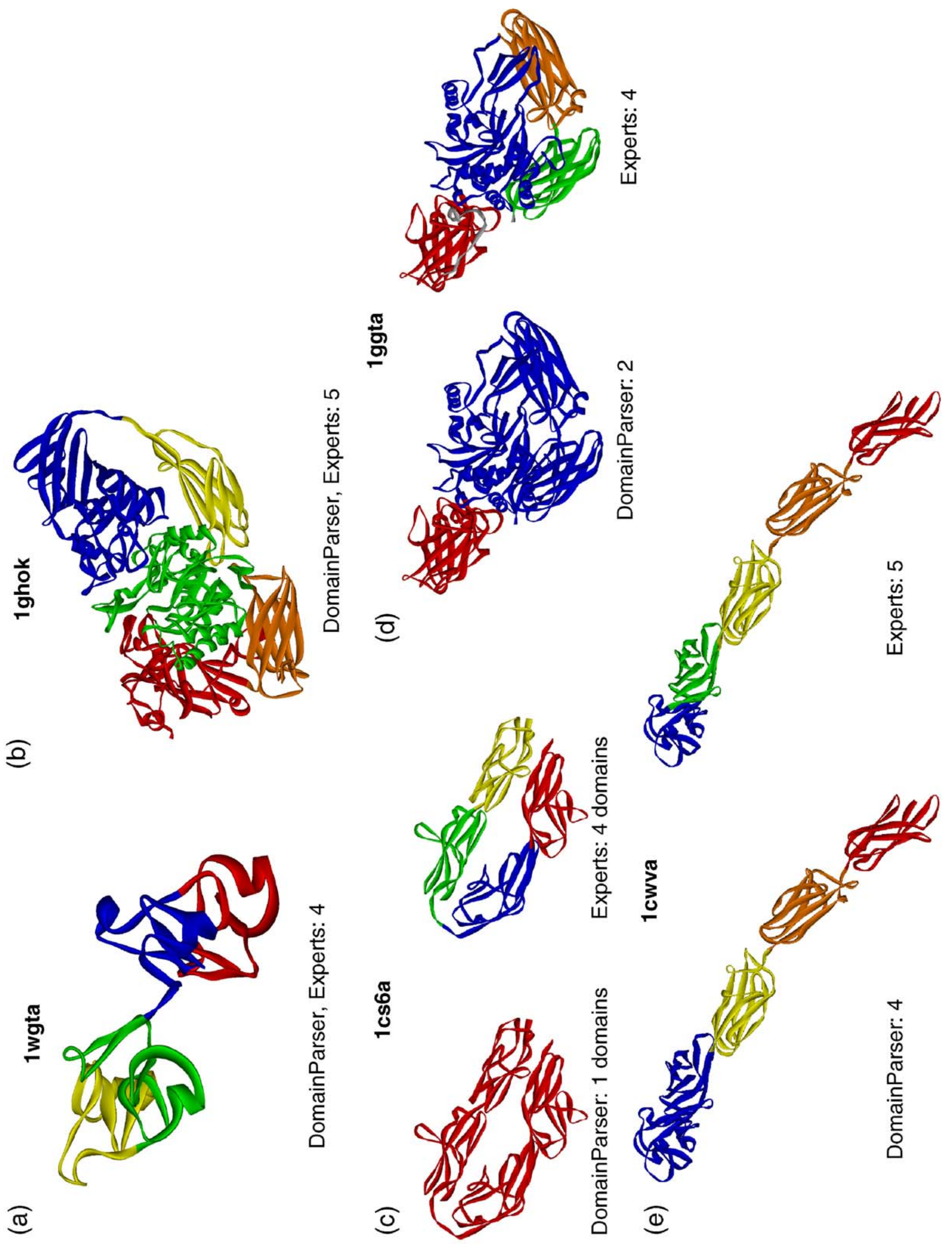


Figure 6 (*legend on next page*)

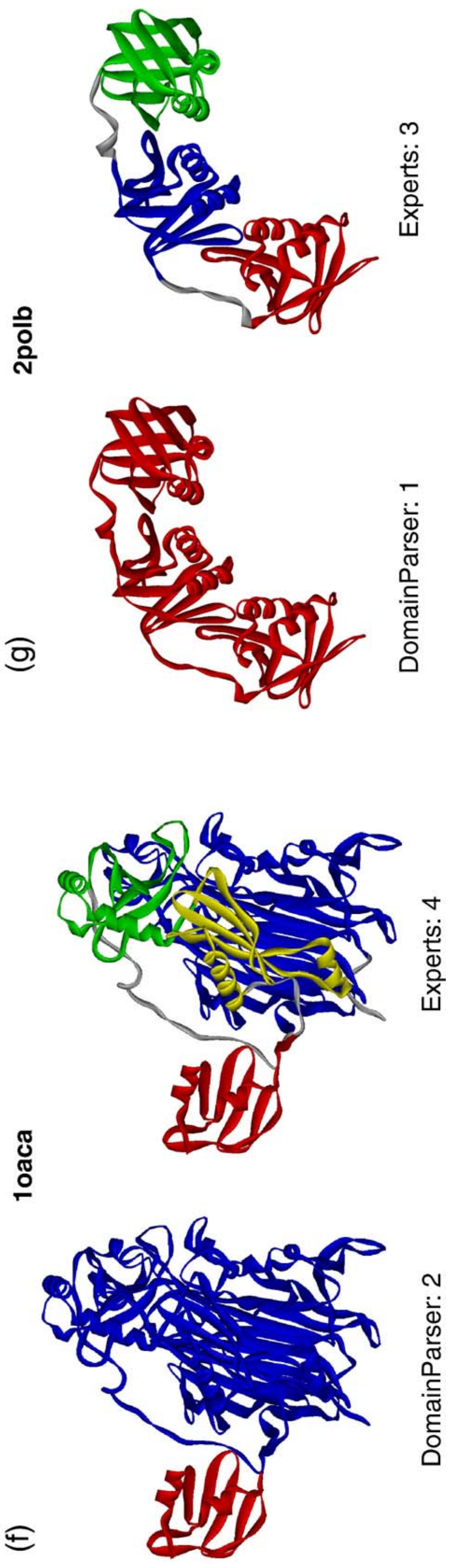


Figure 6. Domain assignments by DomainParser. Gray coloring represents amino acids not assigned to a domain by the given method.

need to be size-dependent. Small domains will have lower inter/intra-domain ratios than larger domains. It may also be based on general class structures: β-structures will have different optimal thresholds than α-helical structures or α/β structures. This latter adjustment based on the structure type may be important in other cases (see below).

Observation. Structures involving one or more β-domains are most problematic for all methods.

An issue. Many of the β-structures are not compact and thus inter-domain density might be indistinguishable from that of intra-domain contact density. For small domains this results simply in the undercutting (some domains do not get separated), but in the case of larger domains (over 60 residues) methods tend either to over-cut or to under-cut structures. Not all β-class domains are difficult. Compact β-structures such as OB-folds, SH-3-type barrels and other barrels are resolved well by all methods (Table 4). Problems exist when there is a complex and convoluted interface between domains, which usually implies non-globular or non-compact domains. Complex domain–domain interfaces occur between two β-domains or between β and α/β domains. The α/β domains often have complex shapes and adding a non-globular β-domain further complicates the problem. Interestingly, β-class domains rarely have problems when in combination with α-class domains, possibly because the different types of packing involved in all-α and all-β leads to a minimal interaction between the domains (Figures 7(b) and 8(j)).

Possible remedy. This is a complex problem and no simple solution can be offered here. Adjusting the threshold of inter/intra- domain contacts for different classes of domains is likely to improve the situation. The bottom line is that density of contacts is a poor discriminator in many β-class domains and in some α/β class domains. Additional features that can be added to an automatic method are the expected size of the domain, given the size of the protein, and the recognition of the overall architectures by the algorithm.

Observation. Large non-compact α-helical structures tend to be partitioned by automatic methods.

Table 5. Integrity of secondary structure criterion

Secondary structure element	DomainParser	PDP	PUU	NCBI	SCOP	CATH
α-Helix (%)	3.3	40.5	25.7	0	4.3	16.7
β-Strand (%)	1.0	7.6	0	0	2.4	4.3

The values are given as a percent of structures in which secondary structure element(s) have been cut by a given method.

Table 6. Summary of performance of each automated method of domain assignment using several key criteria

Method	Overall strength	Weakness	Over-cutting	Under-cutting	Fragmentation	Boundaries–consistency–90	Integrity of secondary structures
PDP	Good assignment of structures with 3 or more domains	Cuts through secondary structures	Middle	Best (least under-cut cases)	Middle	Middle	Worst (most cases of split secondary structures)
PUU		Strong tendency to over-cut structures	Worst (most over-cut cases)	Middle	Worst (domains are highly fragmented)	Middle	Middle
NCBI	Good assignment of structures with 3 or more domains	Difficulty with correct domain boundaries	Middle	Middle	Middle	Worst (poor match with correct boundaries)	Best (no cases of split secondary structures)
DomainParser	Precise domain boundaries overall, good assignment on 1 and 2 domain structures	Strong tendency to under-cut	Best (least over-cut cases)	Worst (most under-cut cases)	Best (domains are optimally fragmented)	Best (good match with domain boundaries)	Middle (actually very few cases of split secondary structures)

An issue. Some α-helical structures are quite sprawling, for example, horseshoe architectures or large orthogonal bundles. These are usually split by all or some automatic methods into multiple domains (Figures 8(e) and 10(d)).

Possible remedy. Contact density clearly fails in such cases. However, contact density works well in most of the α-helical structures. Cataloging large/difficult α-class architectures (similar to that of small β-structures) might be the only feasible solution for automatic methods.

In addition to the issues common to all methods discussed above, below we consider method-specific features.

DomainParser demonstrates the highest propensity among the four automatic methods toward undercutting. That is, predicting fewer domains than predicted by experts. The least problematic structures are large α-class structures such as large orthogonal bundles and up-down bundles as well as large structures in the α/β class, such as complex structures, a horseshoe and three-layer sandwiches (Table 4). The main problem is the failure to continue successful partitioning after the first round, that is, subdividing resulting domains further. Size and compactness of the domain appear as the overriding factors. DomainParser can partition rather complex architectures correctly as long as the resulting domains are either large or very compact (Figure 7(a) and (b)). We suspect that β-strand interactions are contributing greatly to this problem (method has a good success rate for the α-class structures). We also observe that one β-class architecture, that of immunoglobulin-like sandwich, is particularly difficult for DomainParser (Table 4; Figure 7(c), (d) and (e)). This rather simple architecture may epitomize the β-structure issues for this method.

DomainParser performs the most extensive evaluation of the potential domains it assigns. Its failure to further partition large domains may be due to a bias in the post-processing step during which small domains are evaluated and are either granted the status of domains or joined together. Since the multiple criteria used in the post-processing step were trained using SCOP, it is likely that parameters set to favor large domains will under-cut relative to other expert methods just as SCOP tends to do.[2] This will affect the distribution of expected sizes as well as the distribution of the number of fragments used in the post-processing step. Another issue might be improper tuning of β-stand cutting; DomainParser prefers not to cut β-strands between two domains; thus it often keeps large units with interacting β-strands together (Figure 7(c), (d), (f) and (g)). All this points to the possibility that multiple decision-making factors in the post-processing step are not tuned correctly. At the same time DomainParser is the closest among the algorithms to correctly predict

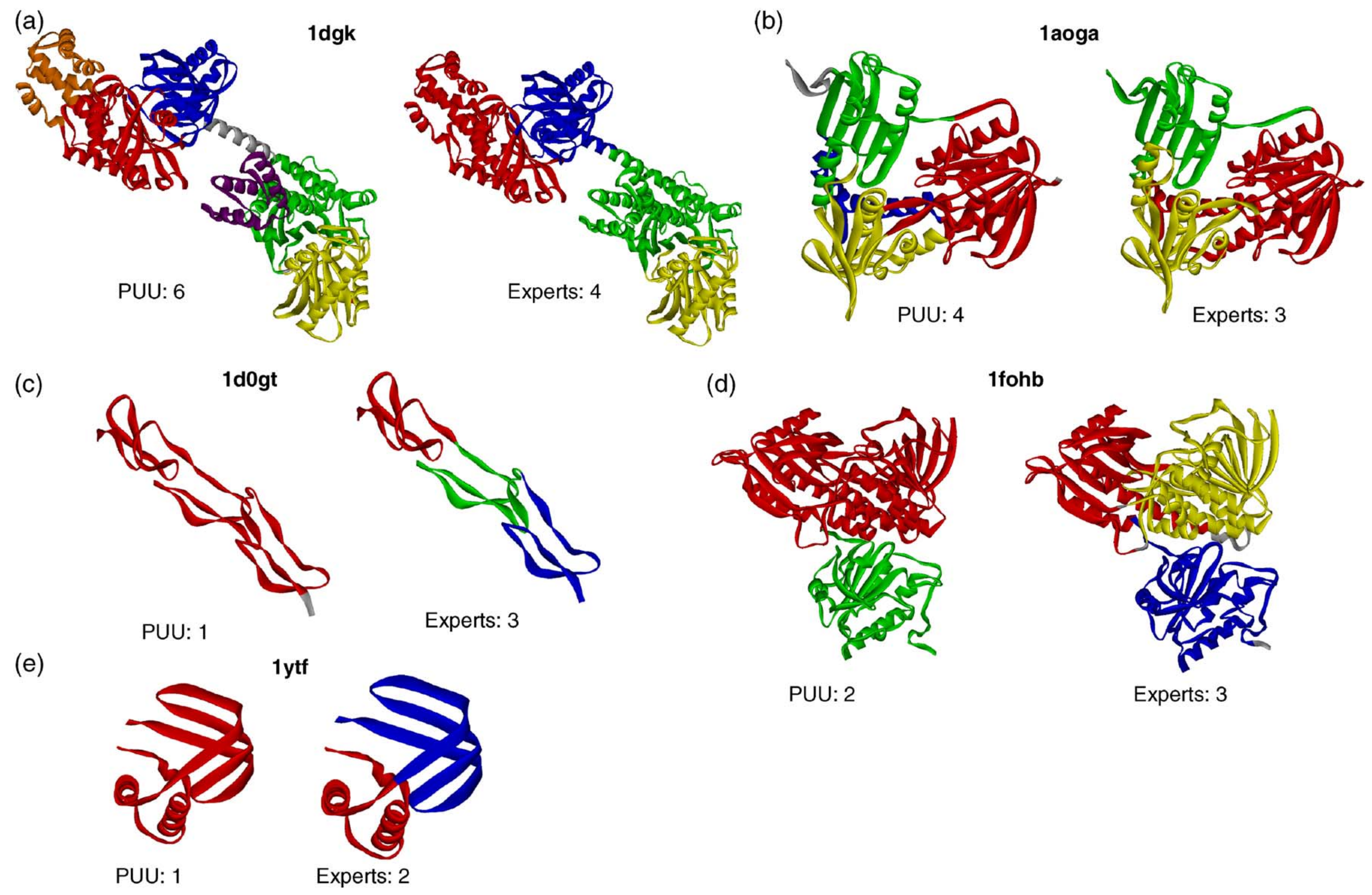


Figure 7. Domain assignments by PUU. Gray coloring represents amino acids not assigned to a domain by the given method.

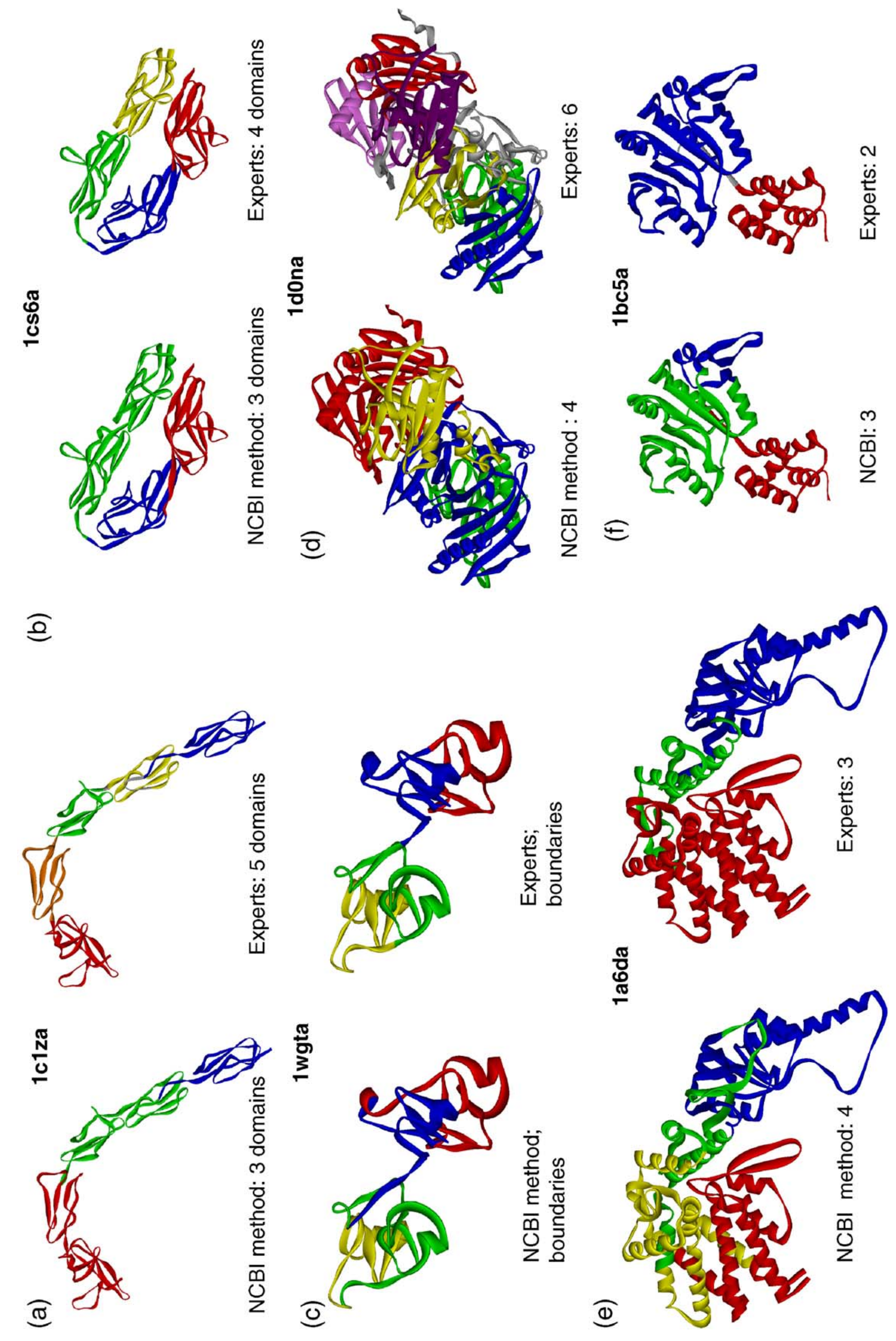


Figure 8 (*legend on next page*)

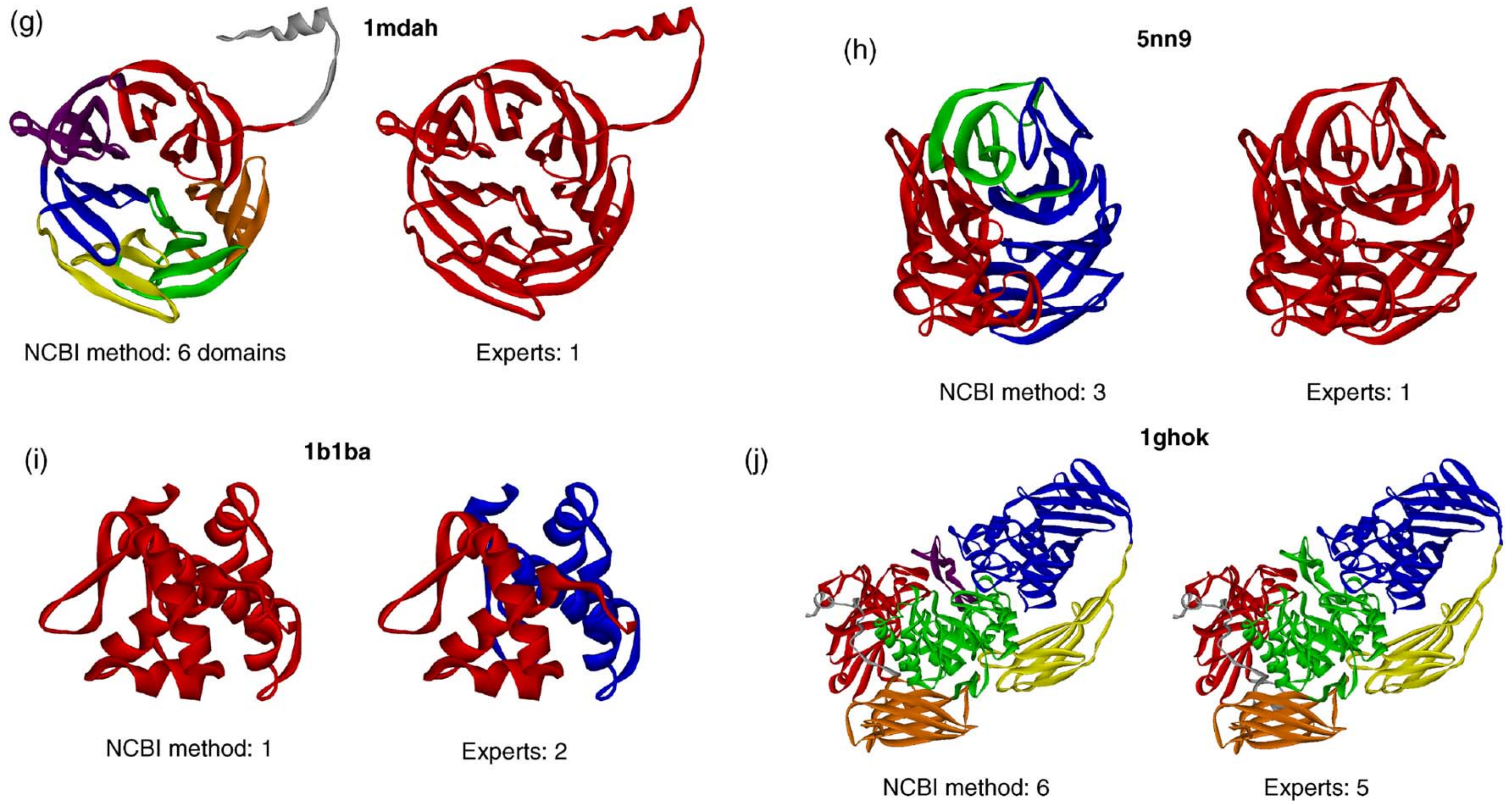


Figure 8. Domain assignment by NCBI. Gray coloring represents amino acids not assigned to a domain by the given method.

the level of domain fragmentation, when compared to expert methods; it also has the most precisely assigned domain boundaries among the four algorithmic methods.

PUU is the oldest method and has the worse performance among the four methods analyzed. It has a strong tendency to over-cut, i.e. predict more domains than experts (Figure 7(a) and (b)). Striving to achieve compact domains, the method frequently assembles domains out of non-contiguous fragments. Sometimes the number of fragments per domain, *s*, is too large to be evolutionarily sensible (Figure 2(a)). Furthermore, the compactness factor appears to confer unrealistically high significance. PUU does not have constraints to prevent splitting α-helices between domains, thus it frequently cuts through helices. However, its rules about integrity of β-sheets, which form "highly cooperative networks," sometimes leave PUU unable to partition all-β structures or structures with a significant fraction of β-sheets (Figure 7(c), (d) and (e)).

NCBI is the most balanced method in its approach as it produces a similar number of under-cut and over-cut errors. When only the number of domains is considered, its overall performance is just trailing that of PDP, the best performing method. Some of NCBI's under-cut errors are due to the inability of the method to cut through secondary structure, an essential feature for any algorithm. In the cases where domains are connected within secondary structure elements (other than a loop) NCBI fails to separate domains, for example, 1c1za (Figure 8(a)) and 1ds6a (Figure 8(b)). In addition a rule concerning placement of domain boundaries in the middle of the loop (1wgta; Figure 8(c)) differs from expert methods (1wgta; Figure 8(c)). The latter feature is detrimental when considering boundary consistency. It performs worst in terms of correct placement of domain boundaries (Figure 4). Another situation of under-cutting involves structures with many domains as in the case of 1d0na (Figure 8(d)). In addition to the need to cut through β-strands, there is the further complication of a convoluted interface.

While NCBI uses an approach similar to PUU (according to the authors of the method), its performance is quite different qualitatively and quantitatively from PUU. PUU nearly always errs in the direction of over-cutting, while NCBI is balanced. Also NCBI performance is highly superior to that of PUU. The heuristics used by NCBI are surprisingly simple compared to those used by PUU. This may indicate that it is not the main principle of domain decomposition that is important, but rather the set of heuristics implemented in the post-processing step that affect the performance of the method. Moreover it appears the simpler the rules the higher the success. For example, PUU easily sacrifices the integrity of secondary structures to achieve compactness of the domain, in the case of NCBI the ratio of compactness/integrity of secondary structure is better balanced. Yet this ratio is not optimal as the method sometimes over-cut large α-structures (1a6da, Figure 8(e); 1bc5a, Figure 8(f)) as well as large β-structures (1mdah, Figure 8(g); 5nn9, Figure 8(h)), while it undercuts smaller α-structures (1b1ba, Figure 8(i)). One of the rules is NCBI does not allow inclusion into a domain of a secondary structure that forms insufficient contacts with the rest of the domain. This too causes cases of over-cut as in 1ghok (1ghok, Figure 8(j)).

PDP has superior performance among the four methods. Its approach is surprisingly simple, yet it is able to achieve good results overall and impressive results on structures with three or more domains. Its overall tendencies are similar to those of PUU. It tends to over-cut structures in order to achieve compactness/globularity of the resulting domains and does not consider secondary structure integrity. In fact this is the only method that does not have any rules with regards to secondary structure elements. We speculate that not taking into account integrity of β-sheets helps PDP to partition multi-domain structures so successfully (1fnma, Figure 9(a); 1d0na; Figure 9(b)). The flip side of disregarding secondary structure integrity is the excessive over-cutting of α-helical structures (1crxa, Figure 9(c); 1aoga, Figure 9(d)) as well as splitting along highly interacting β-sheets (1bc5a, Figure 9(e); 2bpa, Figure 9(f)). The rare cases of under-cut occur in very compact structures such as 1rl2a (Figure 9(g)). The tendency to produce non-contiguous domains is quite moderate in spite of the lack of explicit rules. During the post-processing step the domains are evaluated using the criteria of compactness and minimum contact between domains. If some of the resulting domains fail the criteria, PDP attempts to join them together using the compactness criteria. This feature might contribute to the success of PDP, as it allows an excessive cutting of the structure during the early step, which helps in the case of multi-domain structures, yet it manages to curb many cases of over-cuts by joining small structures into one domain during the post-processing step.

Conclusions

This work addresses several related topics: (1) generation of a comprehensive benchmark dataset for structural domain assignment which adequately covers known structural space; (2) detailed benchmarking of some existing automatic methods using the new benchmark and an extensive set of criteria; and (3) analysis of the performance of the methods in light of the algorithms underlying each method.

It appears that each of the analyzed algorithms (with the exception of PUU, which is inferior to the rest) has its strengths and weaknesses. Often strength in one area is balanced by a weakness in another area. We also observe that different methods have complementary strengths, suggesting the possibility of a hybrid approach to domain assignment. Thus, a meta-method can be developed which uses several of the existing domain assignment methods. Choice of the best method in each case will

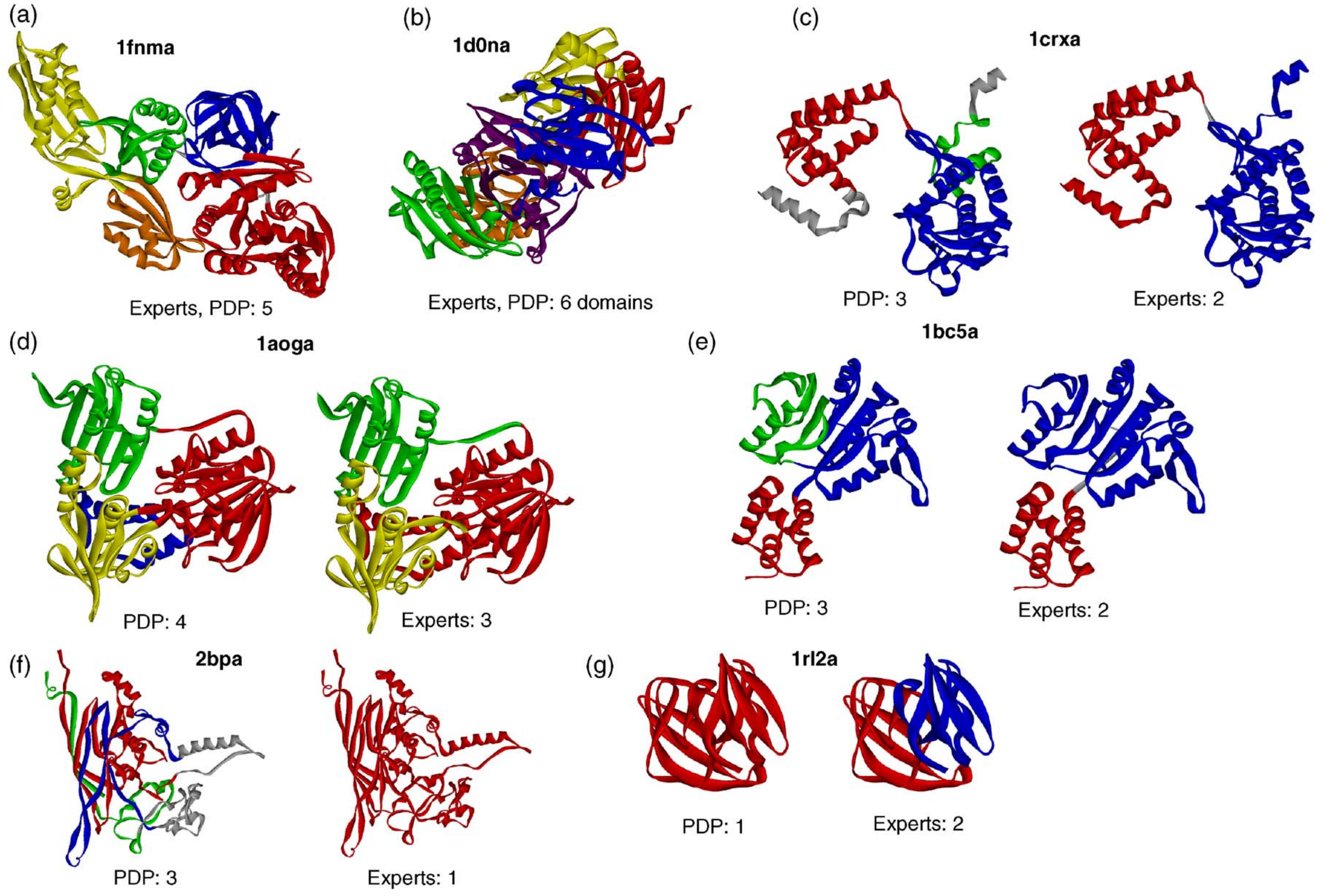


Figure 9. Domain assignments by PDP. Gray coloring represents amino acids not assigned to a domain by the given method.

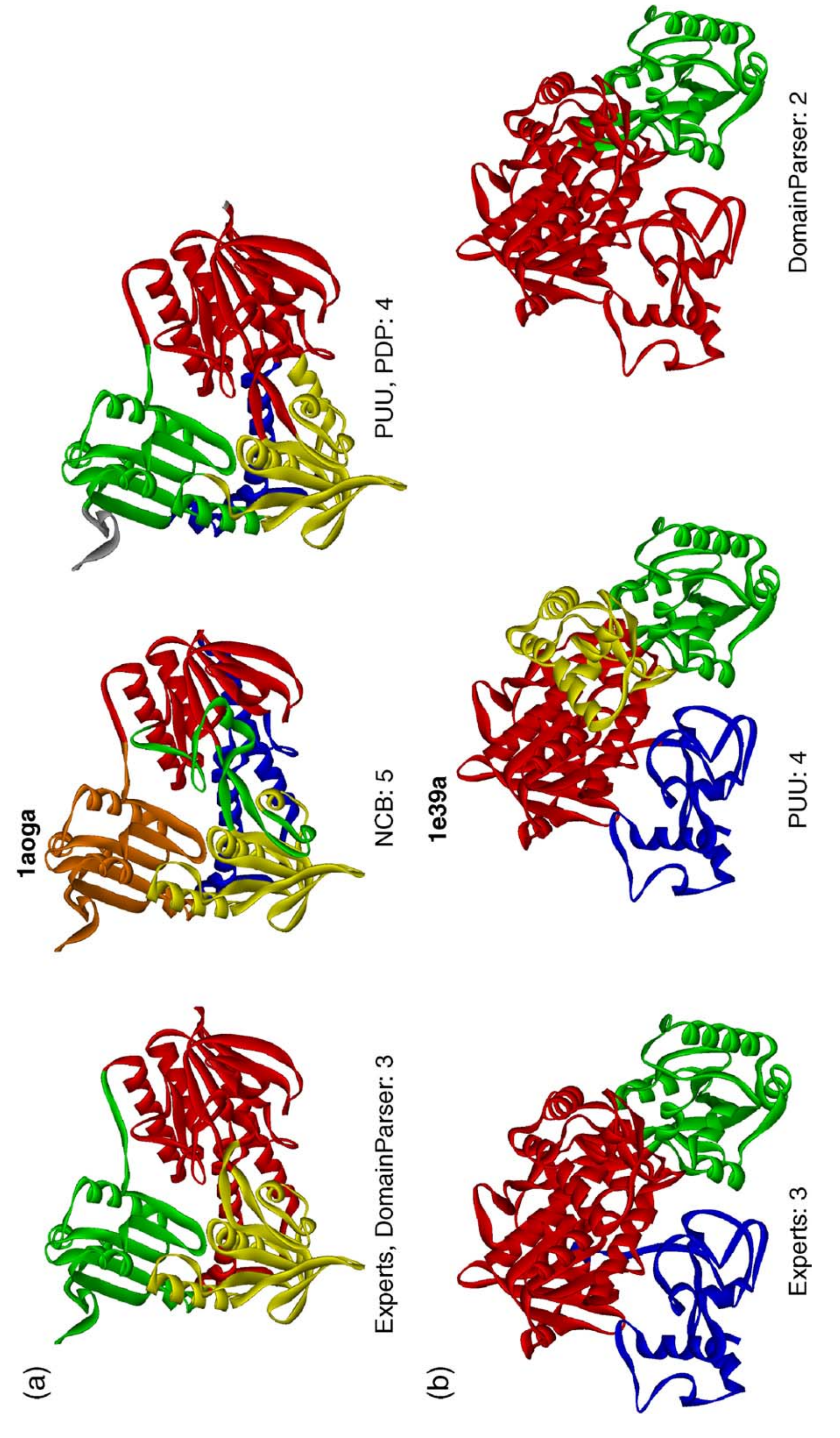


Figure 10 *(legend on page 586)*

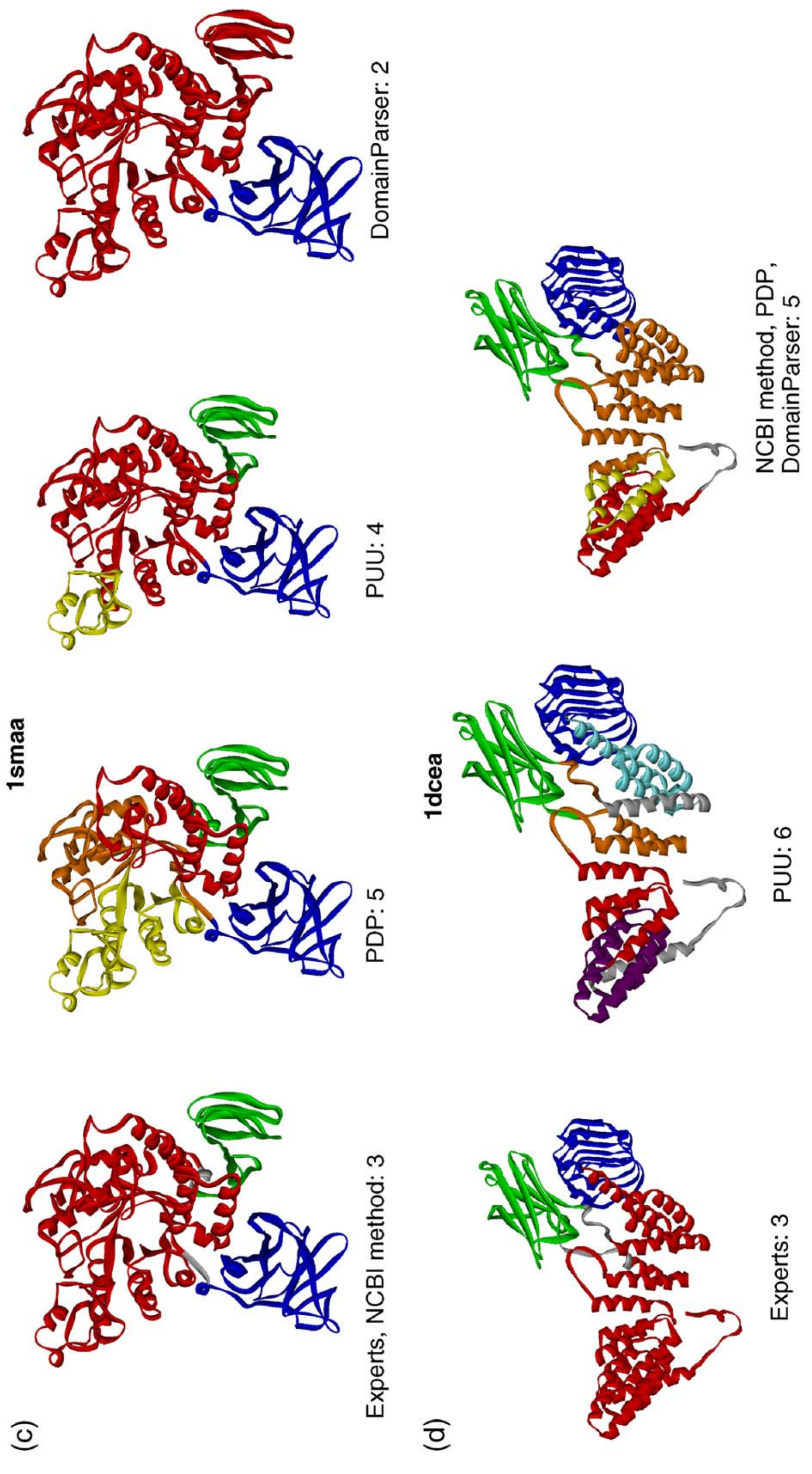


Figure 10 *(legend on next page)*

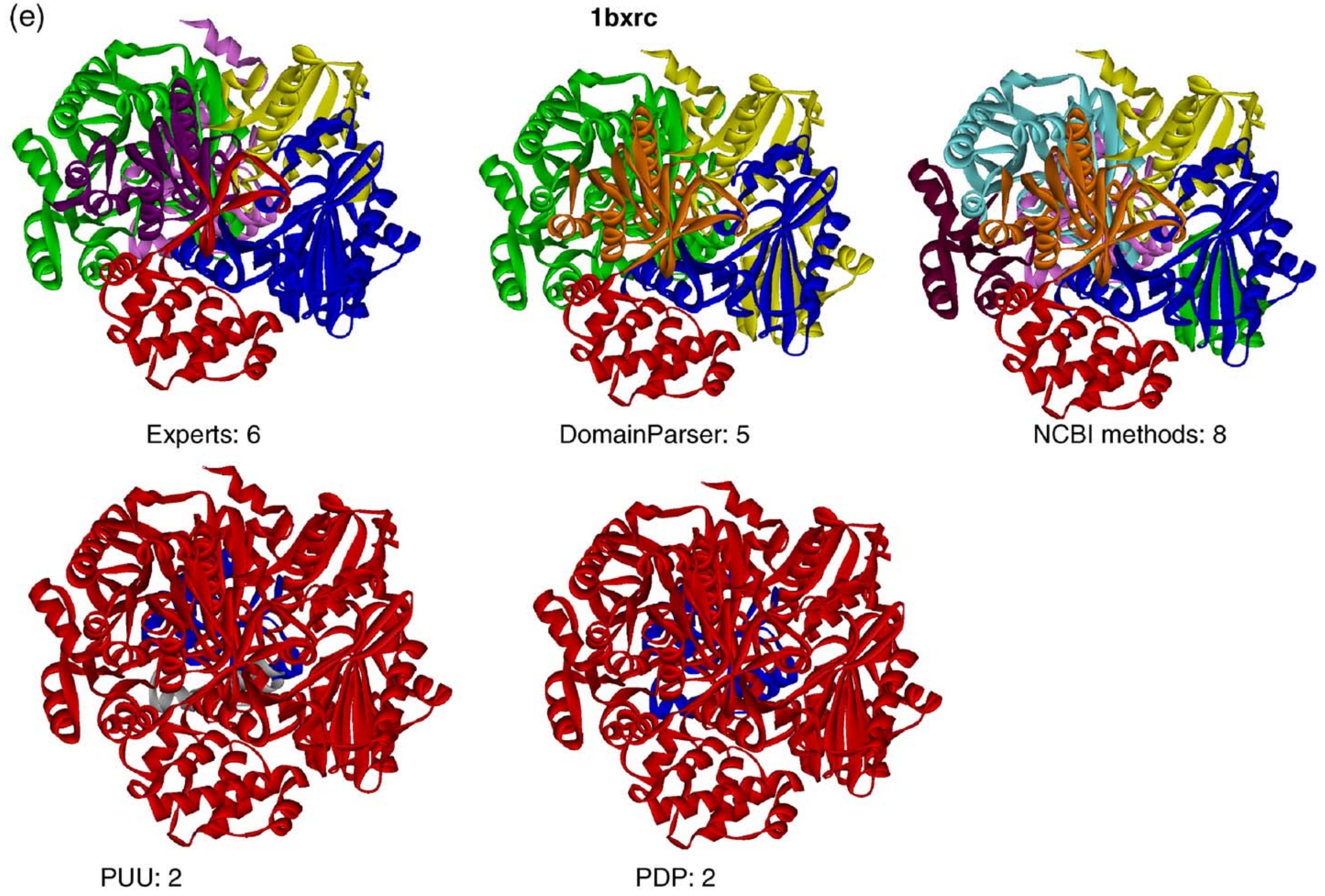


Figure 10. Examples of chains with difficult/complex architectures. Gray coloring represents amino acids not assigned to a domain by the given method.

Table 7. Results of curated data

CATH-SCOP consensus on topology groups	AUTHORS agree with CATH-SCOP consensus	Authors disgree with CATH-SCOP consensus; number of domains assigned by AUTHORS										Number of chains for which no relevant info was found
		1	2	3	4	5	6	7	8	9	10	
2-domain	144	4	144	11	-	-	-	-	-	-	-	20
3-domain	55	-	8	55	2	4	-	-	-	-	-	4
4-domain	8	-	-	-	8	4	-	-	-	-	1	-
5-domain	5	-	-	-	-	5	-	-	-	-	-	-
6-domain	2	-	-	-	-	-	2	-	2	-	-	-

Information provided by the authors of crystallographic or NMR structures. Structures are organized into topology groups, as described in the text.

depend on the size of the protein (which defines the expected number of domains) as well as the topologies involved. Additional types of information currently becoming available from the PDB, such as ligand-binding sites and protein–protein interaction data,[15] may be added to such a meta-method approach.

Our analysis of multi-domain proteins argues that the main difficulty is finding an optimal threshold for inter/intra- domain contact densities. This difficulty is closely linked with the proportion of β-strands in the structure as well as with the size of the resulting domains. Tuning these parameters may further improve performance of existing or new algorithms. Yet if one uses the contact density as a chief parameter in the structure decomposition, as one usually does with automatic methods, it is practically impossible to avoid structure decomposition errors. This is because the contact density by itself is not a completely consistent parameter. Thus, if a method tries to avoid over cutting structures, it inevitably will under-cut some other structures and *vise versa*. The issue of inconsistency appears in the cases of discontinuous domains as well: none of the automatic methods are able to achieve correct (i.e. relatively low) level of domain fragmentation without sacrificing the accuracy of domain assignments. This understanding further strengthens the case for development of a meta-method.

The difficulties of decomposing protein into meaningful structural units is encountered even by expert methods[2] if to a lesser degree than by automatic methods. We observe that while both expert and automatic methods may disagree on the number of domains within a structure (or fragments within a domain), in the majority of cases the partitioning by different methods is similar with the exception that some methods partition the structure further into smaller units. Thus, the basic principles on where domain partitions fall are quite clear, what is not clear is whether to apply a certain partitioning on not. Human expertise employs a myriad of algorithms integrating prior experience, common sense, topological sensibility and other features: thus it comes with consistent answers more often than the algorithms, which after all rely primarily on the contact density data. However, different levels of partitioning the structure and consequently, different sizes and complexities of the resulting domains may reflect a broader and more flexible nature of structural domains. To an evolutionary structural biologist a sensible definition of domain is a recurrent unit in different structures and organisms; to a structural biologist a domain may simply represent a compact folding unit of the structure. Yet from a functional point of view, the domain may be defined by the part of the structure involved in binding, for example. In 10–20% of the cases the above definitions of the domains will be different, thus they will require a different level of partitioning; yet all make sense in the right biological context. Such a context-dependent approach to structural domain definitions has been missing, but might prove fruitful. A logical "reconciliatory" approach might be for the algorithm to provide several progressive levels of domain partitioning and allowing the user to choose the appropriate level of resolution. The first case of such progressive decomposition has been implemented by Berezovsky *et al.*[16] (unfortunately the method was not available for complete benchmarking at the time) and we hope to see this approach gaining ground in the near future.

Methods

Construction of the benchmark dataset

Consensus of expert methods was constructed using CATH v 2.5.1, SCOP 1.65 and AUTHORS assignments. AUTHORS assignments of domains are defined by the authors of crystallographic structures. This information was manually compiled by us after inspection of the literature. Chains for which SCOP and CATH assign identical number of domains were collected using a Perl script, with CathDomainDefinitionFileV.2.5.1 and dir_cla_scop_text.1.65 as input files. There were a total of 20,844 chains. The entire set of chains was then partitioned into groups with identical topology. The CATH definition of topology group was used for this purpose. Chains in each group had the same class, architecture and topology, but possibly different homology. Each chain was placed in at least one group. If a chain contained more than one domain it was placed in multiple groups. Based on the list of all topology groups, a list of existing domain combinations was generated and one AUTHORS assignment collected as a representative of each domain combination.

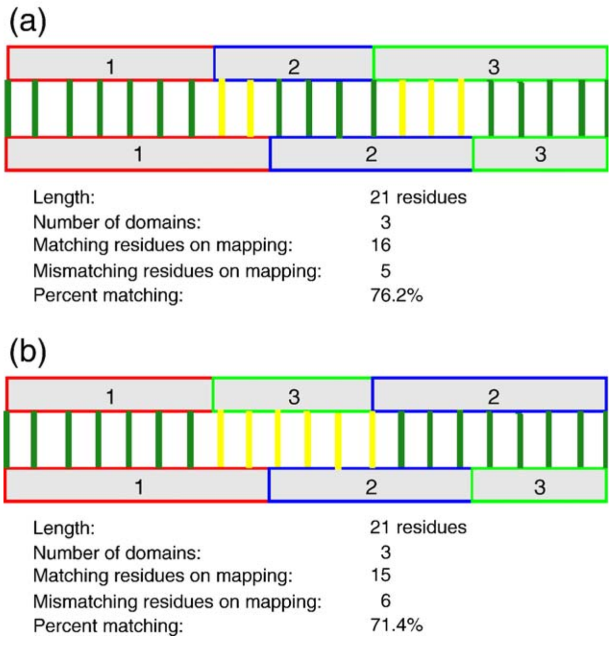


Figure 11. (a) Calculation of Boundaries_consistency90. Each horizontal bar represents how a method assigns domains, each domain is given a number to map to the other method. Each vertical bar represents a residue. Green bars represent matching residues, while yellow bars represent mismatching residues. In this mapping, 76.2% of residues in the chain have been assigned to the same domain by both methods. This number will change when the mapping of domains change in order to find the highest percent matching residues. (b) A rearrangement of the mapping of domains. The method of determining percent matching residues is the same as in (a). In this mapping, the percent residues matching is 71.4%; this is lower than in (a) and therefore (a) would be taken as a better match.

This step involved tedious manual analysis. For each topology group the domain assignment by the authors of the structure was sought using the primary citation recorded by the PDB database (the structure often contains multiple chains; we extracted information for the relevant chain in the structure). If information regarding structural domain was unavailable, we checked the reference publication for the next structure containing the relevant chain. A given topology combination was analyzed until all structures were exhausted or the assignment of domains was found. In 24 cases of chains containing two or three domains, information about domain boundaries could not be inferred, either because the primary PDB citation was incorrect or more frequently because the cited reference did not contain domain information (Table 7). For each structure the following information was recorded during manual curation: number of domains, boundaries of domains, CATH classification of domains, names of domains as given by authors, and a reference to the paper(s) from which information was extracted. If AUTHORS domain assignment (based on number of assigned domains only) corresponded to that of CATH and SCOP, the chain was included in the benchmark dataset. The disagreements between AUTHORS and SCOP-CATH consensus is documented in Table 7. The above analysis was performed for multi-domain chains only. For single-domain chains data were extracted from our earlier benchmark (Benchmark_1). The same criteria of consensus among all methods and one representative per homology were applied. We selected a homology as opposed to topology level of representation due to the poverty of distinct topologies for one-domain proteins. Nevertheless, we did not use more than three one-domain chains that belonged to the same topology group.

The resulting dataset, Benchmark_2, contains 315 chains. Additional details on Benchmark_2 dataset can be found in the Results and Discussion. The Benchmark_3 dataset was constructed by removing 49 chains from Benchmark_2. The eliminated chains were ones with conflicting domain boundary assignments using a stringency of 90% (see below). The resulting Benchmark_3 has 266 chains with 106 one-domain chains, 108 two-domain chains, 45 three-domain chains, seven four-domain chains and five five-domain chains.

Benchmarking automatic methods

For automatic methods

We ran PDP and DomainParser locally. We obtained PUU assignments from PUU Domain Definitions version 3.1 beta[b] For the 11 chains that were missing from this file a local version of the PUU algorithm was run and the pertinent domain information, which contained the number of domains and residue numbers, was extracted using a Perlscript. For NCBI, results were taken from MMDB[c]. The results were converted from an ASN.1 format and provided to us by T. Madej (personal communications).

For expert methods

CATH domain information was extracted from the CathDomainDescriptionFile v. 2.5.1 available form the CATH website[d]. SCOP v.1.65 was used. AUTHORS domains are defined by authors of the structures. These were manually compiled by us after inspection of the literature and provided in the format provided by CATH.

By virtue of the benchmark construction, all expert methods should agree on domain number, this was verified using a Perl script. Under cross-examination we found five chains with no consensus among automatic or expert methods. These were removed from Benchmark_2 resulting in 315 chains.

Evaluating assignments of automatic methods

Domain assignments from each method as well as part of the automatic analysis that followed were performed using a modified version of domain_assign, a C++ analysis package written previously and changed to fit our purposes (Figure 12). Visual inspection and presentation of domain assignments was performed using ViewerPro v. 5.0.0.0[e].

The fraction of correctly assigned domains using the number of domains as a criteria is defined as (number of correctly assigned chains/total number of chains) × 100. A

[b] Holm, L. DaliDomainDictionary.
[c] NCBI. MMDB.
[d] http://cathwww.biochem.ucl.ac.uk/latest/releases.html
[e] Accelrys. ViewerPro.

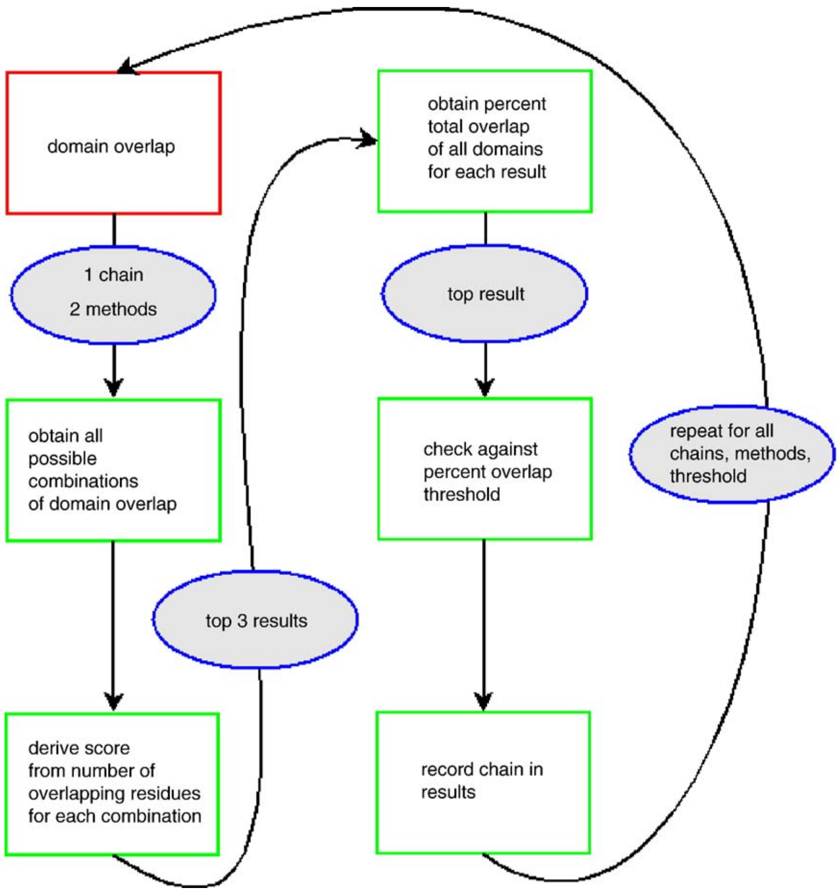

domain overlap
1 chain
2 methods
obtain all possible combinations of domain overlap
top 3 results
derive score from number of overlapping residues for each combination
obtain percent total overlap of all domains for each result
top result
check against percent overlap threshold
record chain in results
repeat for all chains, methods, threshold

Figure 12. Determining domain overlap. Each method has each chain compared to an expert method chosen at the start. This is then checked for domain overlapping by all possible combinations of domains. This is done for all chains for all methods at thresholds of accuracy ranging from 30% to 95%.

correctly assigned chain is defined as a chain for which expert consensus and the current automatic method assign the same number of domains (Figure 1). One expert method, AUTHORS, was used to represent expert consensus.

The number of fragments per domain for each method was obtained by extracting fragment information using a Perl script from the results of each method. The number of domains with one or more fragments per domain was summed by number of fragments for all automatic methods (Figure 2(a)). Expert methods were averaged together and presented in a similar fashion using [(sum of AUTHORS domains with X fragments per domain) + (sum of CATH domains with *X* fragments per domain) + (sum of SCOP domains with *X* fragments per domain)]/(number of expert methods), where *X* is the number of fragments per domain being examined.

The average fragments per domain was determined for all automatic methods and as an average of all expert methods, and is defined as (sum of all fragments/sum of all domains). Expert methods were averaged together by summing the results the average fragments per domain calculation and dividing by the total number of expert methods (Figure 2(b)). These calculations were performed by a Perl script.

The average fragments per domain was determined for all automatic methods and as an average of all expert methods, and is defined as (sum of all domains/sum of all chains). Expert methods were averaged together by summing the results of the average domains per chains and dividing by the total number of expert methods (Figure 2(b)).

Calculating domain overlap

The Boundaries_consistency_90 criterion was performed using a C++ program by identifying when methods agreed on the domain number criterion. The domains assigned by these methods were then mapped to each other, then the percent similarity of residues in each domain computed with the following: [(sum of all residues that map to the same domain on each method)/(total number of residues) × 100%]. In order to ensure that the best mapping between domains assigned by the two methods was found, each possible combination of mapping between domains was computed. The mapping with highest score was used when calculating domain overlap (Figure 11). This calculation was done for each method and for every chain Benchmark_2 and Benchmark_3. Minimum percent matching residues was set starting at a value of 30%, then incrementing by 5% up to 95% (Figure 4); 90% was chosen to be the threshold accuracy at which domain overlap analysis was performed.

Acknowledgements

We are grateful to the authors of the automatic methods for their help in use of their methods. In particular we are thankful to Liisa Holm for directing us to the correct version of DaliDomainDefinitions database, to Nickolai Alexandrov for support of PDP, to Juntao Guo for customizing DomainParser code for our use and to Tom Madej for converting MMDB data for us. We are in debt to Christine Vogel for insightful discussions, to Allen Farquhar for assistance with the C++ code, to Joe Mulvaney for analyzing secondary structure elements in the context of domain boundaries and to Sergey Kushch for his assistance with image processing. This work was supported by the RCSB PDB.

References

1. Chothia, C., Gough, J., Vogel, C. & Teichmann, S. A. (2003). Evolution of the protein repertoire. *Science*, **300**, 1701–1703.
2. Veretnik, S., Bourne, P. E., Alexandrov, N. N. & Shindyalov, I. N. (2004). Toward consistent assignment of structural domains in proteins. *J. Mol. Biol.* **339**, 647–678.
3. Crippen, G. M. (1978). The tree structural organization of proteins. *J. Mol. Biol.* **126**, 315–332.
4. Rose, G. D. (1979). Hierarchic organization of domains in globular proteins. *J. Mol. Biol.* **134**, 447–470.
5. Wodak, S. J. & Janin, J. (1981). Location of structural domains in protein. *Biochemistry*, **20**, 6544–6552.
6. Jones, S., Stewart, M., Michie, A., Swindells, M. B., Orengo, C. & Thornton, J. M. (1998). Domain assignment for protein structures using a consensus approach: characterization and analysis. *Protein Sci.* **7**, 233–242.
7. Islam, S. A., Luo, J. & Sternberg, M. J. (1995). Identification and analysis of domains in proteins. *Protein Eng.* **8**, 513–525.
8. Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B. & Thornton, J. M. (1997). CATH–a hierarchic classification of protein domain structures. *Structure*, **5**, 1093–1108.
9. Murzin, A. G., Brenner, S. E., Hubbard, T. & Chothia, C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* **247**, 536–540.
10. Guo, J. T., Xu, D., Kim, D. & Xu, Y. (2003). Improving the performance of DomainParser for structural domain partition using neural network. *Nucl. Acids Res.* **31**, 944–952.
11. Xu, Y., Xu, D. & Gabow, H. N. (2000). Protein domain decomposition using a graph-theoretic approach. *Bioinformatics*, **16**, 1091–1104.
12. Alexandrov, N. & Shindyalov, I. (2003). PDP: protein domain parser. *Bioinformatics*, **19**, 429–440.
13. Holm, L. & Sander, C. (1994). Parser for protein folding units. *Proteins: Struct. Funct. Genet.* **19**, 256–268.
14. Madej, T., Gibrat, J. F. & Bryant, S. H. (1995). Threading a database of protein cores. *Proteins: Struct. Funct. Genet.* **23**, 356–369.
15. Deshpande, N., Addess, K. J., Bluhm, W. F., Merino-Ott, J. C., Townsend-Merino, W., Zhang, Q. *et al.* (2005). The RCSB Protein Data Bank: a redesigned query system and relational database based on the mmCIF schema. *Nucl. Acids Res.* **33**, D233–D237.
16. Berezovsky, I. N. (2003). Discrete structure of van der Waals domains in globular proteins. *Protein Eng.* **16**, 161–167.

Edited by Michael J. E. Sternberg

(Received 6 January 2006; received in revised form 23 May 2006; accepted 24 May 2006)
Available online 22 June 2006

Note added in proof: It was brought to our attention that we omitted to provide an important reference: SCOP domain assignments were taken from ASTRAL (Chandonia, J. M., Hog, G., Walker, N. S. Lo Conte, L., Koehl, P., Levitt, M. & Brenner, S. E. (2004). *Nucl. Acids Res.* **32**, D189–D192).