To appear in the proceedings of the *Computational Systems Bioinformatics Conference (CSB2008)*

MATCHING OF STRUCTURAL MOTIFS USING HASHING ON RESIDUE LABELS AND GEOMETRIC FILTERING FOR PROTEIN FUNCTION PREDICTION

Mark Moll [1] and Lydia E. Kavraki [1,2,3]

[1] *Department of Computer Science, Rice University, Houston, TX 77005, USA,*
[2] *Department of Bioengineering, Rice University, Houston, TX 77005, USA,*
[3] *Structural and Comp. Biology and Molec. Biophysics, Baylor College of Medicine, Houston, TX 77005, USA*
Email: {mmoll,kavraki}@cs.rice.edu

There is an increasing number of proteins with known structure but unknown function. Determining their function would have a significant impact on understanding diseases and designing new therapeutics. However, experimental protein function determination is expensive and very time-consuming. Computational methods can facilitate function determination by identifying proteins that have high structural and chemical similarity. Our focus is on methods that determine binding site similarity. Although several such methods exist, it still remains a challenging problem to quickly find all functionally-related matches for structural motifs in large data sets with high specificity. In this context, a structural motif is a set of 3D points annotated with physicochemical information that characterize a molecular function. We propose a new method called LabelHash that creates hash tables of n-tuples of residues for a set of targets. Using these hash tables, we can quickly look up partial matches to a motif and expand those matches to complete matches. We show that by applying only very mild geometric constraints we can find statistically significant matches with extremely high specificity in very large data sets and for very general structural motifs. We demonstrate that our method requires a reasonable amount of storage when employing a simple geometric filter and further improves on the specificity of our previous work while maintaining very high sensitivity. Our algorithm is evaluated on 20 homolog classes and a non-redundant version of the Protein Data Bank as our background data set. We use cluster analysis to analyze why certain classes of homologs are more difficult to classify than others. The LabelHash algorithm is implemented on a web server at http://kavrakilab.org/labelhash/.

1. INTRODUCTION

High-throughput methods for structure determination have greatly increased the number of proteins with known structure in the Protein Data Bank[1]. Determining the function of all these proteins would greatly impact drug design. Unfortunately, functional annotation has not kept up with the pace of structure determination. Sequence-based methods are an established way for automated functional annotation[2–5], but sequence similarity does not always imply functional similarity and vice versa. Structural analysis allows for the discovery of similar function in proteins with very different sequences and even different folds[6]. For an overview of current approaches in sequence- and structure-based methods see Refs. 7 and 8.

Structure-based methods can be divided into several categories, such as methods that compare fold similarity[9, 10], methods that model pockets and clefts[11–13], and search algorithms based on active sites and templates (see section 2). The combination of structural and phylogenetic information can be used to identify residues that are of structural or functional importance[14–16]. Several web servers exist that use a combination of several sequence- and structure-based methods[17, 18] to provide an aggregate of information.

The method in this paper falls in the template search category. We will describe a new method for partial structure comparison. In partial structure comparison, the goal is to find the best geometric and chemical similarity between a set of 3D points called a *motif* and a subset of a set of 3D points called the *target*. Both the motif and targets are represented as sets of labeled 3D points. A motif is ideally composed of the functionally most-relevant residues in a binding site. The labels denote the type of residue. Motif points can have multiple labels to denote that substitutions are allowed. Any subset of the target that has labels that are compatible with the motif's labels is called a *match*. The aim is to find statistically significant matches to a structural motif. Our method preprocesses, in a fashion that borrows ideas from a well-known technique called geometric hashing[19], a background database of targets such as a non-redundant subset of the Protein Data Bank. It does this in such a way that we can look up in constant time partial matches to a motif. Using a variant of the previously described match augmentation algorithm[20], we obtain complete matches to our motif. The nonparametric statistical model developed in Refs. 21 and 22 corrects for any bias introduced by our algorithm. This bias is introduced by excluding matches that do not satisfy certain geometric constraints for efficiency reasons.

The contributions of this paper are as follows. Our new method is based on hashing of residue labels and geometric constraints, an approach that proves to be efficient, highly sensitive, and highly specific. It further improves the already high specificity of our previous work. It removes the requirement of needing an ordering of the importance of the points of the motifs. Using cluster analysis, we provide a more complete picture of match results and we illustrate the difficulty of matching certain functional homologs. Last but not least, our approach can be easily adapted to use different motif types or incorporate different constraints. Although not discussed in detail in this paper, we can optionally include partial matches or multiple matches per target in the match results. Before we will describe our method, we will first give an overview of related methods.

2. RELATED WORK

Over the years several algorithms have been proposed for the motif matching problem. In its generality, this problem has a chemical, a geometric, and a statistical component. First, points in our motif need to be matched with chemically compatible points. This can translate into simply matching the same residue types, but can also be defined in terms of a more general classification of physicochemical properties[23, 24]. Geometrically, we want to solve the partial structure comparison problem: find all correspondences between a motif and groups of points in the targets that are chemically compatible. Solving issues associated with the high complexity of the problem are discussed in Ref. 25. Most existing methods employ heuristics to find only matches that are close under the Least Root Mean Square Deviation (LRMSD) metric, since these matches are most likely functionally related to the motif. This brings us to the statistical component of the problem: there is no universal LRMSD threshold that can be used to separate functional homologs from other matches, and thus statistical analysis is needed to decide whether a match is functionally related to a motif and unlikely to occur due to chance.

In table 1 we have summarized some selected related work that we will discuss in more detail below. A direct comparison of our work with other methods is challenging for several reasons: (1) there are several ways to represent structural motifs, (2) most of the methods included in the table solve a slightly different version of the problem discussed in this paper, and (3) for most systems there is no freely available or web-accessible implementation with which we could perform experiments similar to our own.

Geometric hashing[19, 31, 32] is a technique to preprocess the targets that will be used for matching and create index tables that facilitate fast matching. These tables only need to be computed once for a set of targets. They are based on geometric characteristics. One has to carefully pick the geometric constraints to balance the potentially enormous storage requirements with completeness of the matching phase. The application of geometric hashing to motif matching was first introduced in Ref. 19 and has been refined in subsequent years. TESS[27] is an algorithm that uses geometric hashing to match structural motifs. By focusing on a specific class of motifs (catalytic triads), TESS can create space-efficient hashing tables. More recent work on geometric hashing[31] uses several "pseudo-centers" per residue to represent physicochemical properties to achieve more accurate matching.

In Ref. 23 a graph-based approach is used. Residues are represented by a pair of pseudo-atoms. The pseudo-atoms are the vertices of the graph, and edges are defined by the distances between them. The matching is then solved by solving the subgraph isomorphism problem[33]. In Ref. 34 distance constraints on C_α and C_β atoms are used to guide a depth-first search for matches. Unlike much of the previous work, this paper also introduced a statistical model to determine the significance of a match. Matching results were fitted to an extreme value distribution and allowed for matching of catalytic triads

Table 1. Overview of selected related work.

Name	Physicochemical information	Geometric algorithm	Statistical model	Demonstrated application
ASSAM[23]	pairs of pseudo-atoms per residue	subgraph isomorphism	—	catalytic triads
FEATURE[26]	supervised learning of many physicochemical and geometric features		nonparametric model, Bayesian scoring	ATP-binding, S-S sites, Mg^{2+} binding sites in RNA
TESS[27]	all atoms of selected residues	geometric hashing	—	His-based catalytic triads
Jess[28]	user-defined constraints on atoms	constraint satisfaction + match augmentation	mixture of two Gaussians	HTH motifs
PINTS[29]	reduced # res., 1 pseudocenter per res.	depth-first search w/distance constraints	extreme value dist. on weighted RMSD	catalytic triads, salt bridges, S-S sites
DRESPAT[30]	C_α's, C_β's, and functional atoms	graph based on distance constraints, max. complete subgraph detection	significance estimated from algorithm parameters & output	detection of many motifs (e.g., catalytic triads, EF-hand)
SiteEngine[31]	pseudocenters	geometric hashing	—	finding and comparing functional sites
MASH[20]	evolutionary importance, residue-labeled C_α's	match augmentation	nonparametric model	matching motifs of $\sim$5–15 residues against large data sets
LabelHash [this paper]	residue-labeled C_α's	hash tables of res. labels + match augmentation	nonparametric model	matching motifs of $\sim$5–15 residues against large data sets

and zinc fingers[29]. More recently, in Ref. 30 a graph-based method was described that automatically detects repeating patterns in subgraphs of graph representations of proteins. This is reduced to a graph clique enumeration problem, a well-known, very difficult problem in general, but by taking advantage of the structure of the underlying data, this method can avoid the worst-case complexity.

The FEATURE algorithm[26] takes a radically different approach to matching. It uses supervised learning to characterize the active sites of proteins. Many attributes can be defined and the learning algorithm will automatically learn the salient features. More recently, this algorithm has been applied to ATP-binding and disulfide bond-forming sites[35] and magnesium binding sites in RNA[36]. Although in is original form the FEATURE algorithm worked directly on structural data, later work showed that it is able to construct structural motifs from sequence-based motifs[37]. The FEATURE algorithm is accessible through a public web server[38]. The representation of motifs is very different, making comparison with other methods challenging.

In Ref. 39 a parametrized statistical model is proposed to determine the significance of the LRMSD of a match. The model parameters are obtained by fitting the model to the data. This model is part of the PINTS server[29], which uses a distance constraint-based matching algorithm similar to the one described in Ref. 34. The PINTS server used to allow matching against a non-redundant subset of the PDB, but at the time of writing this option was no longer available, making a comparison with our method difficult. In Ref. 28 a more general matching framework is proposed, where user-defined constraints can be associated with a number of residues. The residues and constraints together form a template. A mixture of two Gaussians is used to model the distribution of the LRMSD's of matches. The same template-based approach was successfully applied to finding DNA-binding proteins that contain the helix-turn-helix (HTH) motif[40]. This last work also showed that for finding HTH matches, 3D templates could be used to detect similarity between many different HTH motifs, while a sequence-based approach based on Hidden Markov Models could not.

Recent work on template matching[41] argues in favor of using a heuristic similarity measure rather than LRMSD to rank matches. This similarity measure is a function of the number of overlapping atoms and a residue mutation score. It is shown to eliminate many false positives in certain cases. This paper introduces so-called reverse templates, which are conceptually similar to geometric hashing's notion of reference sets.

In Ref. 20 the MASH matching algorithm is introduced. It is based on a depth-first search that finds matches to motif points in order of decreasing importance ranking. Our approach is most compatible with this algorithm. In our algorithm we preprocess the targets to speed up matching, remove the need for importance ranking, and improve specificity. Further improvements can be made to the MASH algorithm by explicitly representing cavities[42] and by creating composite motifs in case several instances of a functional site are known[43].

3. METHODS

We are interested in matching a structural motif against a set of targets. The structural motif is defined by the backbone C_α coordinates of a number of residues and (optionally) allowed residue substitutions for each motif residue which are encoded as labels. Previous work[44, 39, 20, 14] has established that this is a feasible approach. There is no fundamental reason why other points cannot be used as well.

The method presented below is called LabelHash. It builds hash tables for n-tuples of residues that occur in a set of targets. In spirit the method is reminiscent of the geometric hashing technique[19], but the particulars of the approach are different. The n-tuples are hashed based on the residues' labels. Each n-tuple has to satisfy certain geometric constraints. Using this table we can look up partial matches of size n in constant time. These partial matches are augmented to full matches with an algorithm similar to MASH[20]. Compared to geometric hashing[19], our method significantly reduces the storage requirements. Relative to MASH, we further improve the specificity. Also, in the LabelHash algorithm it is no longer required to use importance ranking of residues to guide the matching. In our previous work, this ranking was obtained using Evolutionary Trace (ET) information[45]. The LabelHash algorithm was designed to improve the (already high) accuracy of MASH and push the envelope of matching with only very few geometric constraints. For this work we wanted motifs to be as general as possible to allow for future extensions and to facilitate motif design through a variety of methods. The input should be easy to generate from "raw data" such as PDB files, and the output should be easy to post-process and visualize. Although the ideal of functional annotation is full automation, an exploratory process of iterative and near-interactive motif design and refinement will be extremely valuable. Our simple-to-use and extensible LabelHash algorithm can be a critical component of this process. The LabelHash algorithm consists of two stages: a preprocessing stage and a stage where matches are computed from the preprocessed data.

3.1. Preprocessing Stage

The preprocessing stage has to be performed only once for a given set of targets. Every motif can be matched against the same preprocessed data. During the preprocessing stage we aim to find possible candidate partial matches. This is done by finding all n-tuples of residues that satisfy certain geometric constraints. We will call these n-tuples *reference sets*. All valid reference sets for all targets are stored in a hash map, a data structure for key/value pairs that allows for constant time insertions and lookups (on average). In our case, each key is a sorted n-tuple of residue labels, and the value is a list of reference sets that contain residues with these labels in any order. So for any reference set in a motif we can instantly find all occurrences in all targets. Notice that in contrast to geometric hashing[19] we do not store copies of the targets for each reference set, which allows us to store many more reference sets in the same amount of memory.

In our current implementation the geometric constraints apply to the C_α coordinates of each residue, but there is no fundamental reason preventing other control points from being used instead. We have defined the following four constraints:

- Each C_α in a reference set has to be within a distance $d_{\text{maxmindist}}$ from its nearest neighboring C_α.
- The maximum distance between any two C_α's within a reference set is restricted to be less than d_{diameter}.
- Each residue has to be within distance d_{maxdepth}

from the molecular surface. The distance is measured from the atom closest to the surface.
- At least one residue has to be within distance $d_{\text{maxmindepth}}$ from the surface.

The first pair of constraints requires points in reference sets to be within close proximity of each other, and the second pair requires them to be within close proximity of the surface. The distance parameters that define these constraints should be picked large enough to allow for at least one reference set for each motif that one is willing to consider, but small enough to restrict the number of seed matches in the targets. One would perhaps expect that the storage requirements would be prohibitively expensive, but, in fact, in the experiments described in section 4 we used very generous settings, and the storage used was still very reasonable.

3.2. Matching Stage

For a valid reference set in a motif, we look up the matching reference sets in the hash table. Next, we run a variant of the match augmentation algorithm[20] that consists of the following steps. First, it solves the residue label correspondence between a motif reference set and the matching reference sets stored in the hash map. (If more than one correspondence exists, we will consider all of them.) Next, we augment the match one residue at a time, each time updating the optimal alignment that minimizes the LRMSD. If a partial match has an LRMSD greater than some threshold ε, it is rejected. For a given motif point, we find all residues in a target that are within some threshold distance (after alignment). This threshold is for simplicity usually set to ε. The ε is set to be sufficiently large (7Å in our experiments) so that no interesting matches are missed. The value ε also affects the computation of the statistical significance of a match. It can be shown that for a motif of n residues our statistical model computes the *exact* p-value of matches with LRMSD less than $\varepsilon/\sqrt{n}$, i.e., their p-value would not change if no ε threshold was used [22, 21]. For example, for a 6-residue motif and $\varepsilon = 7$Å, the p-values of all matches within 2.3Å of the motif are exact.

The algorithm recursively augments each partial match with the addition of each candidate target residue. The residues added to a match during match augmentation are *not* subject to the geometric constraints of reference sets. In other words, residues that are not part of a reference set are allowed to be further from each other and more deeply buried in the core. For small-size reference sets, the requirement that a motif contains at least one reference set is therefore only a very mild constraint. Nevertheless, as we will see in the next section, our approach is still highly sensitive and specific.

For a given motif, we generate all the valid reference sets for that motif. Any of these reference sets can be used as a starting point for matching. However, those reference sets that have the smallest number of matching reference sets in the hash map may be more indicative of a unique function. Reference sets with a large number of matches are more likely to be common structural elements or due to chance. We could exhaustively try all possible reference sets, but for efficiency reasons we only process a fixed number of least common reference sets. Note that the selection of reference sets as seed matches is based only on frequency. In contrast, in our previous work, only one seed match was selected based on importance ordering frequently based on evolutionary importance[20]. Because of the preprocessing stage it now becomes feasible to expand matches from many different reference sets. The hash map files have embedded within them a “table of contents,” so that during matching only the relevant parts of the hash map need to be read from disk.

The matching algorithm is flexible enough to give users full control over the kind of matches that are returned. It is possible to keep partial matches that match at least a certain minimum number of residues. This can be an interesting option for larger motifs where the functional significance of each motif point is not certain. In such a case, a 0.5Å LRSMD partial match of, say, 9 residues, might be preferable over a 2Å complete match of 10 residues. With partial matches, the matches can be ranked by a scoring function that balances the importance of LRMSD, and the number of residues matched. One can also choose between keeping only the LRMSD match per target or all matches for a target, which may be desirable if the top-ranked matches for targets have very similar LRMSD’s. Keeping partial matches or multiple matches per target complicates the determination of the statistical significance of each match. This is an issue we plan to investigate in future work. Finally, the number of motif reference sets that the algorithm uses for match augmentation can also be varied. Usually most matches are found with the first couple reference sets, but occasionally a large number of reference sets need to be tried before the LRMSD match for each target is found.

4. RESULTS

4.1. Data sets

LabelHash was tested on a diverse set of previously identified motifs. The motifs we used in our experiments are listed in table 2. Some were determined experimentally, others were determined using the Evolutionary Trace (ET) method[45]. More information on the function of these motifs and how they were obtained can be found in Refs. 20 and 42. Although the performance of the matching algorithm depends critically on the motif, the focus in this paper is on the motif matching *method* and not on motif *design*. Any motif of a similar type can be used by our method. For each motif we have listed the residue sequence numbers, followed by the 1-letter residue name and possible substitutions. The substitutions were in some cases determined using ET, but any reasonable set is accepted (sometimes experiments or intuition give the substitutions). It is important to note that our algorithm is "neutral" with respect to how a motif is obtained; importance ranking or other very specific information on the motif is not required.

Table 2. Motifs used in experiments

PDB ID	Residue ID's with alternate labels
1acb	$42^{GSN}, 57, 58^{SKV}, 102, 194^{QE}, 195, 214^{AT}$
1ady	$81^{D}, 83, 112^{S}, 130^{D}, 264^{L}, 311^{NKQ}$
1ani	$51^{A}, 101^{E}, 102, 166^{CS}, 331^{G}, 412^{NQ}$
1ayl	$249, 250, 251, 253, 254, 255$
1b7y	$149^{GA}, 178^{Q}, 180^{T}, 206^{ER}, 218, 258^{NY}, 260^{Y}$
1czf	$178, 180, 201, 256^{H}, 258, 291$
1did	$25, 53, 56, 93, 136, 182$
1dww	$194, 346, 363, 366, 367^{F}, 371, 376^{D}$
1ep0	$53^{TA}, 61^{A}, 64, 73, 90, 172$
1ggm	$188^{T}, 239^{T}, 341, 311^{L}, 359^{S}, 361^{A}$
1jg1	$97^{DNQ}, 99, 101^{AL}, 160^{NS}, 179^{VI}, 183^{NE}$
1juk	$53, 89, 91, 233, 182, 110$
1kp3	$106, 139, 202^{S}, 286, 288, 331$
1kpg	$17, 72, 74, 75, 76, 200$
1lbf	$51, 56, 57, 89, 91, 112, 159, 180, 211, 233$
1nsk	$12^{RL}, 13, 52^{HL}, 105^{H}, 115, 118^{P}$
1ucn	$12, 13, 92, 105, 115, 118$
2ahj	$53, 120, 127, 190, 193, 196$
7mht	$80, 81, 85^{T}, 119^{L}, 163, 165$
8tln	$120^{WL}, 143^{A}, 144^{VI}, 157^{SL}, 231^{L}$

The targets consisted of a non-redundant snapshot of the Protein Database (PDB), taken on February 21, 2008. We used the automatically generated 95% sequence identity filtered version of the PDB. Each chain was inserted separately in the hash map. This resulted in roughly 18,000 targets. Molecular surfaces were computed with the MSMS software[46]. We chose to use reference sets of size 3. The following parameter values were used for the reference sets:

$$d_{\text{maxmindist}} = 16\text{Å}, \quad d_{\text{diameter}} = 25\text{Å},$$
$$d_{\text{maxmindepth}} = 1.6\text{Å}, \quad d_{\text{maxdepth}} = 3.1\text{Å}.$$

These values were chosen such that the motifs in table 2 contained at least one reference set of size 3. They are very generous in the sense that most motifs contain many reference sets. If reference sets of more than 3 residues are used, the values of the distance parameters need to be increased to guarantee that each motif contains at least one reference set. The advantage of larger reference sets is that we instantly match a larger part of a motif. However, increasing these values also quickly increases the number of reference sets in the targets. So the number of reference sets to perform match augmentation on will also quickly increase. Finally, the storage required for the hash tables grows rapidly with reference set size. After the preprocessing phase the total hash map size given the settings described above was 32GB (split into several files).

4.2. Matching results

The results of matching the motifs from table 2 against the targets is shown in table 3. We evaluated the performance using the PDB entries with the corresponding Enzyme Classification (EC) code or corresponding Gene Ontology (GO) molecular function term as the set of positive matches. Typically, there is more than one GO molecular function term associated with one PDB entry. We picked the most

Table 3. Matching results with a p-value of 0.001.

	Enzyme classification		Gene Ontology		
PDB ID	TP	FP	TP	FP	time (s)
1acb	87.50% (28)	0.08% (13)	—	—	27541
1ady	100.00% (22)	0.07% (13)	68.00% (17)	0.08% (14)	10268
1ani	75.61% (62)	0.06% (11)	—	—	12673
1ayl	100.00% (19)	0.07% (12)	—	—	3006
1b7y	40.00% (8)	0.07% (12)	40.00% (4)	0.07% (12)	15744
1czf	100.00% (21)	0.04% (7)	100.00% (13)	0.06% (9)	1078
1did	100.00% (152)	0.02% (2)	100.00% (108)	0.02% (2)	181
1dww	88.94% (209)	0.04% (5)	95.31% (183)	0.04% (5)	1635
1ep0	100.00% (39)	0.05% (8)	100.00% (21)	0.05% (8)	2308
1ggm	81.82% (9)	0.07% (12)	33.33% (5)	0.07% (13)	12620
1jg1	100.00% (17)	0.06% (11)	100.00% (13)	0.07% (13)	44982
1juk	100.00% (12)	0.06% (10)	—	—	1211
1kp3	100.00% (36)	0.06% (10)	100.00% (35)	0.07% (11)	637
1kpg	84.62% (11)	0.06% (8)	84.62% (11)	0.06% (8)	126
1lbf	100.00% (12)	0.05% (8)	77.78% (7)	0.06% (9)	2650
1nsk	72.91% (148)	0.00% (0)	—	—	7128
1ucn	81.77% (166)	0.01% (2)	—	—	851
2ahj	35.90% (14)	0.06% (10)	33.33% (11)	0.07% (12)	420
7mht	90.91% (10)	0.08% (10)	—	—	2130
8tln	95.08% (58)	0.08% (14)	—	—	1989

specific term (i.e., the one with the fewest PDB entries). For some motifs no GO annotation for molecular function is available, which is indicated by a '—'. The true and false positives are given as percentages followed by the absolute number of matches between parentheses. In most cases our method finds close to 100% of all known true positives with a p-value of 0.001, and only very few false positives. Even in absolute terms the number of false positives is very small. For the 1acb motif, which represents the catalytic triad, we only counted α-chymotrypsin as a true positive. This excludes several other members of the corresponding EC class. An additional complication for this motif is that it sometimes spans several (but not all) chains in a complex. In this case we manually separated chymotrypsin from its inhibitor.

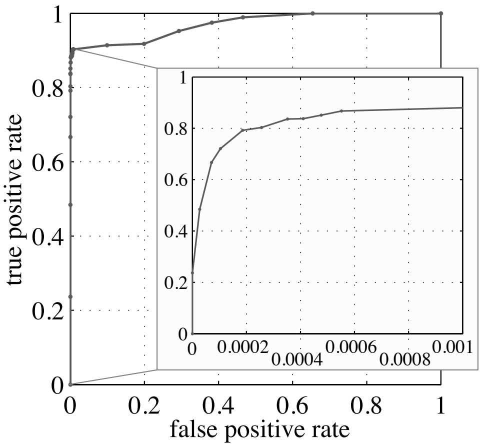
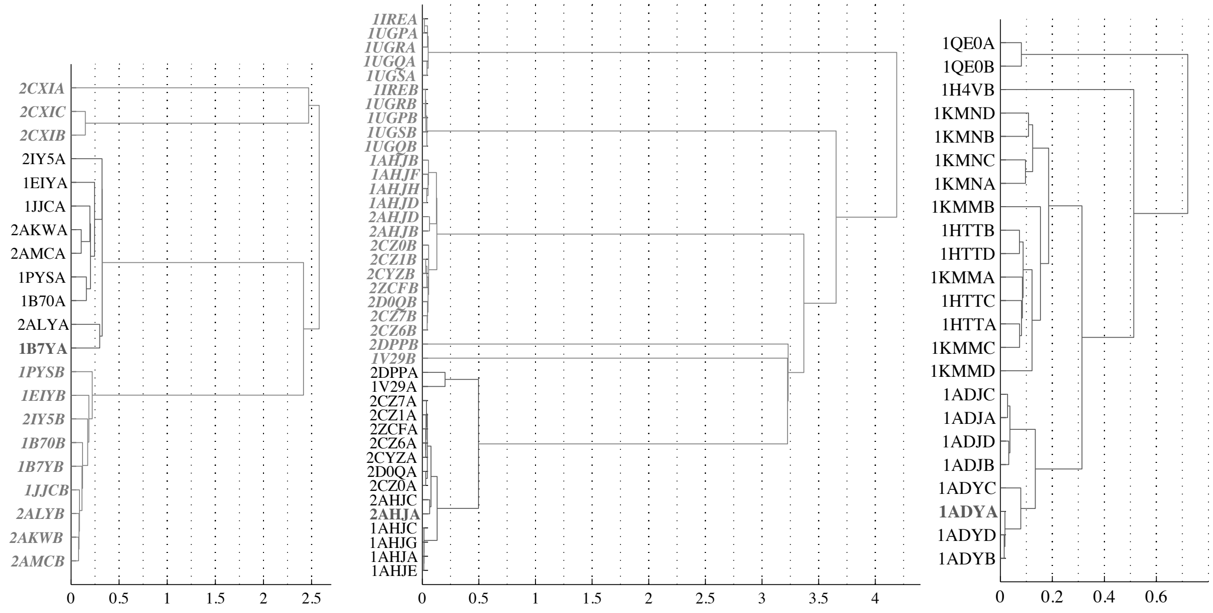


Fig. 1. ROC curve. The true positive rate and false positive rate are averaged over all motifs at a given p-value. The inset plot shows the performance for very small false positive rates.

Figure 1 shows the false positive rate and true positive rate as we vary the p-value. The true positive rate and false positive rate are averaged over all motifs at a given p-value. With MASH, our previous algorithm, we could achieve on average a 83.7% true positive rate at a 0.98% false positive rate. Now, at the same false positive rate, we achieve 90% sensitivity. Or, alternatively, at the same true positive rate, we now achieve a 0.04% false positive rate. The improvement in false positive rate is especially significant. Since in our case the number of targets is so much larger than the number of homologs, a small false positive rate can still mean that the number of false positives is many times larger than the number of true positives. For example, for the 8tln motif the false positive rate went from 9.1% with MASH to 0.08% with LabelHash. In absolute terms, the *number* of false positive matches went from 168 with MASH to 14 with LabelHash. *In almost all cases the number of false positives is now less than the number of true positive matches.*

The p-values of matches are computed using a so-called point-weight correction[22, 21]. This is a sta-



(a) 1b7y, EC 6.1.1.20

(b) 2ahj, EC 4.2.1.84

(c) 1ady, EC 6.1.1.21

Fig. 2. Clustering of matches in EC classes for three motifs. Matches in bold italics are likely to be missed because they are in a cluster that is very different from the cluster that contains the motif (shown in bold).

tistical correction for the bias introduced by only considering matches in a small neighborhood of motif points. While using the neighborhood heuristic typically preserves biologically relevant matches, eliminating biologically irrelevant matches can affect the accuracy of thresholds provided by the statistical models of sub-structural similarity. Statistical models depend on an unbiased representation of matches to yield the most accurate thresholds. During the match augmentation phase of the algorithm we only considered matching points in targets that were up to $\varepsilon = 7$Å away, but other matching points may exist. These other matches tend to be in right-hand side of the RMSD distribution of matches. The existence of these matches can be determined by simply looking at residue frequencies for each target. The point-weight represents these matches in the p-value determination. This can significantly improve the accuracy, especially for small ε. For a relatively large value of $\varepsilon = 7$Å, the effect is relatively small: *with* the point-weight the average sensitivity for the motifs in table 2 is 86.0%, but *without* the point-weight this drops to 82.7%. The specificity is relatively unaffected: it changes from 99.94% (with point-weight) to 99.96% (without). However, if a small $\varepsilon = 3$Å threshold is used, the sensitivity with point-weight is 85.7%, and without point-weight it is 32.9%. Again, specificity is relatively unaffected: 99.94% with point-weight and 99.996% without. The reason one may want to use a small value for ε is that it significantly reduces the runtime. The total time for matching all of the motifs in table 2 can be reduced by almost 60% by changing ε from 7Å to 3Å. The accuracy improvements over MASH observed at $\varepsilon = 7$Å are also observed at smaller ε levels.

To better understand what happens when a homolog is classified as false negative, let us now consider the homolog matches for three motifs. Suppose we take all the homolog matches for a given motif, compute all pairwise LRMSD distance between the matches, and cluster the results based on these distances. We expect that matches that end up in a different cluster than the motif's cluster, are more likely to be misclassified. This is indeed what appears to be the case for our motifs. Figure 2 shows dendrograms computed for three motifs. For the 1b7y motif and corresponding homologs in the EC 6.1.1.20 class of homologs there are two very distinct large clusters consisting of the 'A' and 'B' chains, respectively, and one small cluster for the outlier protein 2cxi. The 'B' chains of enzymes in EC 6.1.1.20 are very different from the 'A' chains. The assigned function for this class is really a property of the complex, rather than a single chain. It is therefore not surprising that the 'A' chains do not match the 'B' chains very well. For the 2ahj motif the situation is more complex (see figure 2(b)). Again, there are very distinct classes, but this time it is not obvious why this is the case. The last example, for 1ady and homologs, shows a dendrogram for a case where our matching algorithm found all homologs. Now all homologs are very close to each other and the clusters are not well-separated. This suggests that cluster analysis on match results can provide additional insight into whether matches are likely to be functionally related to a motif.

The runtime of matching each motif against 18,000 targets in the non-redundant PDB is shown in the last column of table 2. The time is highly variable: it ranges from a couple of minutes to several hours. The variability is due to the size of the motif, the type of residues, and—most importantly—the number of alternate labels. For instance, for the 1jg1 motif the number of alternate labelings for the entire motif is $4\times1\times3\times3\times3\times3 = 324$. Although we do not match each alternate labeling separately, the increased branching factor during match augmentation still exponentially increases the runtime. Compared to MASH, our previous algorithm, the runtime has increased by a factor 5. This is due mostly because LabelHash algorithm performs match augmentation on many reference sets (up to 40 per motif in our experiments), whereas MASH only used one reference set, because its definition of the reference set was based on the availability of importance ranks for the residues. We expect that further parameter optimization and code profiling will allow LabelHash to run at comparable speed, but with superior accuracy. Comparison with other approaches was attempted, but it was impossible to complete due to reasons given in section 2. In particular, the problems solved are not always the same, or it is not possible to translate our motifs, or compare performance results. In an effort to help in solution of this problem in the future, a web server that will enable the community to use our work has been implemented and is accessible at http://kavrakilab.org/labelhash. More demanding users can also download a command line version that offers more options. We have also developed a prototype match visualization plugin for Chimera[47]. It superimposes the selected match with the motif and shows some additional information such as the corresponding EC and GO terms. On demand, additional information from PDBsum[48] is displayed. This will give the user an incredible wealth of information about a match. The ViewMatch plugin is also available at the LabelHash web site.

The runtime is measured by running the matching on a single CPU core of a 2.2GHz Dual Core AMD Opteron processor. Multi-core processors and distributed computing clusters are increasing commonplace, and naturally we would like to take advantage of that. Both the preprocessing stage and the matching stage are trivially parallelized, and a near-linear speed-up with the number of CPU cores can be expected. In the preprocessing phase we divide the targets into a number of groups and build a hash map for each. Each core can be responsible for building a number of hash maps. This requires no communication. Matching can also easily be parallelized. Each core can match a given motif against a set of targets independently. Once matching is finished, the match results can be aggregated into one output file by one of the cores.

5. CONCLUSION AND FUTURE WORK

We have presented LabelHash, a new algorithm for partial structural alignment. It quickly matches a motif consisting of residue positions, and possible residue types to large sets of targets. We have shown that LabelHash achieves very high sensitivity and specificity with 20 motifs matched against a background data set consisting of the non-redundant PDB filtered at 95% sequence identity. Accuracy is further improved due to a nonparametric statistical model that corrects for systematic bias in our algorithm. Typically, the number of false positive matches is much smaller than the number of true positive matches, despite the large number of targets in our background database. This greatly speeds up the analysis of match results. Our algorithm uses only a small number of parameters whose meaning is easy to understand. We have shown that clustering of matches can provide useful clues about the functional similarity between a motif and a match.

Extensibility was an important factor in the design of the LabelHash implementation. Our program is easily extended to incorporate additional constraints or use even conceptually different types of motifs. For instance, matching based on physico-chemical pseudo-centers[23, 24] could easily be incorporated, and we plan to offer this functionality in the future. Input and output are all in XML format, which enables easy integration with other tools or web services. It is also not hard to imagine LabelHash as part of a multi-stage matching pipeline. The matches produced by LabelHash can be given to the next program, which can apply additional constraints to eliminate more false positives. As long as the set of matches produced by LabelHash include all functional homologs, this seems to be a viable strategy. Of course, the output of LabelHash can also easily be passed on to any clustering algorithm (as was done for figure 2) or a visualization front-end.

As mentioned at the end of section 3, our matching algorithm has the capability to keep partial matches and multiple matches per target. This makes the statistical analysis significantly more complicated. Currently, we just disable the p-value computation when either option is selected, but we plan to investigate the modeling of the statistical distribution of these matches.

Acknowledgements

The project upon which this publication is based was performed pursuant to Baylor College of Medicine Grant No. DBI-054795 from the National Science Foundation. LK has also been supported by a Sloan Fellowship. The computers used to carry out experiments of this project were funded by NSF CNS 0454333 and NSF CNS-0421109 in partnership with Rice University, AMD and Cray.

The authors are indebted to V. Fofanov for many useful discussions on the use of statistical analysis and for his comments on LabelHash. They are also deeply grateful for the help of B. Chen and D. Bryant with MASH. This work has benefited from earlier contributions by O. Lichtarge, M. Kimmel, D. Kristensen and M. Lisewski within the context of an earlier NSF funded project. The authors would also like to thank the members of the Kavraki Lab for proof-reading this paper.

References

1. Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000) The protein data bank. *Nucleic Acids Research,* **28**, 235–242.
2. Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990) Basic local alignment search tool. *J. Mol. Biol,* **215**(3), 403–410.
3. Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research,* **22**(22), 4673–4680.
4. Eddy, S. R. (1996) Hidden markov models. *Curr Opin Struct Biol,* **6**(3), 361–365.
5. Finn, R. D., Tate, J., Mistry, J., Coggill, P. C., Sammut, S. J., Hotz, H.-R., Ceric, G., Forslund, K., Eddy, S. R., Sonnhammer, E. L. L., and Bateman, A. (2008) The Pfam protein families database. *Nucleic Acids Res,* **36**(Database issue), D281–8.
6. Hermann, J. C., Marti-Arbona, R., Fedorov, A. A., Fedorov, E., Almo, S. C., Shoichet, B. K., and Raushel, F. M. (2007) Structure-based activity prediction for an enzyme of unknown function. *Nature,* **448**(7155), 775–779.
7. Watson, J., Laskowski, R., and Thornton, J. (2005) Predicting protein function from sequence and structural data. *Current Opinion in Structural Biology,* **15**(3), 275–284.
8. Zhang, C. and Kim, S. H. (2003) Overview of structural genomics: from structure to function. *Current Opinion in Chemical Biology,* **7**(1), 28–32.
9. Holm, L. and Sander, C. (1993) Protein structure comparison by alignment of distance matrices.. *J Mol Biol,* **233**(1), 123–138.
10. Zhu, J. and Weng, Z. (2005) FAST: a novel protein structure alignment algorithm.. *Proteins,* **58**(3), 618–627.
11. Binkowski, T. A., Freeman, P., and Liang, J. (2004) pvSOAR: detecting similar surface patterns of pocket and void surfaces of amino acid residues on proteins. *Nucleic Acids Res,* **32**, W555–W558.
12. Dundas, J., Ouyang, Z., Tseng, J., Binkowski, A., Turpaz, Y., and Liang, J. (2006) CASTp: computed atlas of surface topography of proteins with structural and topographical mapping of functionally annotated residues. *Nucleic Acids Research,* **34**(Web Server issue), W116–W118.
13. Laskowski, R. A. (1995) SURFNET: a program for visualizing molecular surfaces, cavities, and intermolecular interactions. *J Mol Graph,* **13**(5), 323–330.
14. Kristensen, D. M., Ward, R. M., Lisewski, A. M., Chen, B. Y., Fofanov, V. Y., Kimmel, M., Kavraki, L. E., and Lichtarge, O. (2008) Prediction of enzyme function based on 3D templates of evolutionary im-

portant amino acids. *BMC Bioinfomatics,* **9**(17).

15. Glaser, F., Rosenberg, Y., Kessel, A., Pupko, T., and Ben-Tal, N. (2005) The ConSurf-HSSP database: the mapping of evolutionary conservation among homologs onto PDB structures. *Proteins,* **58**(3), 610–617.
16. Chakrabarti, S. and Lanczycki, C. (2007) Analysis and prediction of functionally important sites in proteins. *Protein Science,* **16**(1), 4.
17. Laskowski, R. A., Watson, J. D., and Thornton, J. M. (2005) ProFunc: a server for predicting protein function from 3D structure. *Nucleic Acids Research,* **33**, W89–W93.
18. Pal, D. and Eisenberg, D. (2005) Inference of protein function from protein structure. *Structure,* **13**(1), 121–130.
19. Nussinov, R. and Wolfson, H. J. (1991) Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques.. *Proc Natl Acad Sci U S A,* **88**(23), 10495–10499.
20. Chen, B. Y., Fofanov, V. Y., Bryant, D. H., Dodson, B. D., Kristensen, D. M., Lisewski, A. M., Kimmel, M., Lichtarge, O., and Kavraki, L. E. (2007) The MASH pipeline for protein function prediction and an algorithm for the geometric refinement of 3D motifs. *J. Comp. Bio.,* **14**(6), 791–816.
21. Fofanov, V. Y. Statistical Models in Protein Structural Alignments PhD thesis Department of Statistics, Rice University Houston, TX (2008).
22. Fofanov, V. Y., Chen, B. Y., Bryant, D., Moll, M., Lichtarge, O., Kavraki, L., and Kimmel, M. (2008) Correcting systematic bias caused by algorithmic thresholds in statistical models of protein sub-structural similarity. *BMC Biology Direct,* Submitted.
23. Artymiuk, P. J., Poirrette, A. R., Grindley, H. M., Rice, D. W., and Willett, P. (1994) A graph-theoretic approach to the identification of three-dimensional patterns of amino acid side-chains in protein structures. *Journal of Molecular Biology,* **243**(2), 327–344.
24. Schmitt, S., Kuhn, D., and Klebe, G. (2002) A new method to detect related function among proteins independent of sequence and fold homology. *J Mol Biol,* **323**(2), 387–406.
25. Shatsky, M. The Common Point Set Problem with Applications to Protein Structure Analysis PhD thesis School of Computer Science, Tel Aviv University (June, 2006).
26. Bagley, S. C. and Altman, R. B. (1995) Characterizing the microenvironment surrounding protein sites. *Protein Sci,* **4**(4), 622–635.
27. Wallace, A. C., Borkakoti, N., and Thornton, J. M. (1997) TESS: A geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. application to enzyme active sites. *Protein Science,* **6**(11), 2308.
28. Barker, J. A. and Thornton, J. M. (2003) An algorithm for constraint-based structural template matching: application to 3D templates with statistical analysis. *Bioinformatics,* **19**(13), 1644–1649.
29. Stark, A. and Russell, R. B. (2003) Annotation in three dimensions. PINTS: Patterns in non-homologous tertiary structures. *Nucleic Acids Research,* **31**(13), 3341–3344.
30. Wangikar, P. P., Tendulkar, A. V., Ramya, S., Mali, D. N., and Sarawagi, S. (2003) Functional sites in protein families uncovered via an objective and automated graph theoretic approach. *J Mol Biol,* **326**(3), 955–978.
31. Shulman-Peleg, A., Nussinov, R., and Wolfson, H. J. (June, 2004) Recognition of functional sites in protein structures. *J Mol Biol,* **339**(3), 607–633.
32. Wolfson, H. J. and Rigoutsos, I. (1997) Geometric hashing: an overview. *IEEE Computational Science and Engineering,* **4**(4), 10–21.
33. Ullmann, J. R. (1976) An algorithm for subgraph isomorphism. *J. of the ACM,* **23**(1), 31–42.
34. Russell, R. B. (1998) Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution. *Journal of Molecular Biology,* **279**(5), 1211–1227.
35. Wei, L. and Altman, R. B. (2003) Recognizing complex, asymmetric functional sites in protein structures using a Bayesian scoring function. *J Bioinform Comput Biol,* **1**(1), 119–138.
36. Banatao, D. R., Altman, R. B., and Klein, T. E. (2003) Microenvironment analysis and identification of magnesium binding sites in RNA. *Nucleic Acids Research,* **31**(15), 4450–4460.
37. Liang, M. P., Brutlag, D. L., and Altman, R. B. (2003) Automated construction of structural motifs for predicting functional sites on protein structures.. In *Pacific Symposium on Biocomputing.* pp. 204–215.
38. Liang, M. P., Banatao, D. R., Klein, T. E., Brutlag, D. L., and Altman, R. B. (2003) WebFEATURE: an interactive web tool for identifying and visualizing functional sites on macromolecular structures. *Nucleic Acids Research,* **31**(13), 3324–3327.
39. Stark, A., Sunyaev, S., and Russell, R. B. (2003) A model for statistical significance of local similarities in structure. *Journal of Molecular Biology,* **326**(5), 1307–1316.
40. Jones, S., Barker, J. A., Nobeli, I., and Thornton, J. M. (2003) Using structural motif templates to identify proteins with DNA binding function. *Nucleic Acids Research,* **31**(11), 2811–2823.
41. Laskowski, R. A., Watson, J. D., and Thornton, J. M. (2005) Protein function prediction using local 3D templates. *Journal of Molecular Biology,* **351**(3), 614–626.
42. Chen, B. Y., Bryant, D. H., Fofanov, V. Y., Kristensen, D. M., Cruess, A. E., Kimmel, M., Lichtarge, O., and Kavraki, L. E. (April, 2007) Cavity scaling: Automated refinement of cavity-aware motifs in pro-

tein function prediction. *Journal of Bioinformatics and Computational Biology,* **5**(2), 353–382.

43. Chen, B. Y., Bryant, D. H., Cruess, A. E., Bylund, J. H., Fofanov, V. Y., Kimmel, M., Lichtarge, O., and Kavraki, L. E. (2007) Composite motifs integrating multiple protein structures increase sensitivity for function prediction. In *Comput Syst Bioinformatics Conf.*
44. Yao, H., Kristensen, D. M., Mihalek, I., Sowa, M. E., Shaw, C., Kimmel, M., Kavraki, L., and Lichtarge, O. (Feb, 2003) An accurate, sensitive, and scalable method to identify functional sites in protein structures. *J Mol Biol,* **326**(1), 255–261.
45. Lichtarge, O., Bourne, H. R., and Cohen, F. E. (Mar, 1996) An evolutionary trace method defines binding surfaces common to protein families. *J Mol Biol,* **257**(2), 342–358.
46. Sanner, M. F., Olson, A. J., and Spehner, J. C. (1996) Reduced surface: an efficient way to compute molecular surfaces.. *Biopolymers,* **38**(3), 305–320.
47. Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., and Ferrin, T. E. (2004) UCSF Chimera—a visualization system for exploratory research and analysis. *Journal of Computational Chemistry,* **25**(13), 1605–1612.
48. Laskowski, R. A. (2001) PDBsum: summaries and analyses of PDB structures. *Nucleic Acids Research,* **29**(1), 221–222.