T U M

I N S T I T U T F Ü R I N F O R M A T I K

A Fast Method for Motif Detection and Searching in a Protein Structure Database

Arno Buchner Hanjo Täubig

TUM-I0314
September 03

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-09-I0314-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

A Fast Method for Motif Detection and Searching in a Protein Structure Database

Arno Buchner* Hanjo Täubig†

buchner@in.tum.de taeubig@in.tum.de

Institut für Informatik, Technische Universität München
D-80290 München

Abstract

We present a data structure based on suffix trees that provides fast methods for searching three-dimensional polypeptide structures and for detecting common substructures (motifs) in a protein structure database. Unlike former approaches in this topic our approach does not consider the sequence of amino acids (i.e. the primary structure). Instead of that it uses a coding of the backbone torsion angles which is invariant to translation and rotation of the molecule in the actual coordinate system.

By applying structural queries to a suffix tree that was build for the polypeptides in the Protein Data Bank (PDB) it is shown that the method works in practice and that it might be a very useful tool for future research in proteomics.

1 Introduction

Recently, questions related to the three-dimensional structure of proteins attracted huge interest because the structure is crucial for the function of a protein. Bioinformatics researchers are facing the problem that predicting a protein structure from the sequence of amino acids is quite complicated until now. Computer scientists formalized the problem but did not get useful results due to oversimplification (e.g. using discrete lattice models). Ab initio methods use a thermodynamics approach trying to find the structure with minimum free energy. However, this can only be done correctly for very small instances because of the complexity of the applied algorithms. So the prediction of the correct folding of the amino acid chain is still one of the major open problems of bioinformatics research today.

One way to solve it might be learning as much as possible from the structures that have been determined in the past years by X-ray crystallography and NMR methods and that have been deposited in public databases like the PDB[1]. In this context there are a lot of interesting questions that occur frequently and that must be answered efficiently (amongst others):

*supported by the BMBF within the BFAM research project
(Bioinformatics for the Functional Analysis of Mammalian Genomes)
†supported by DFG grant Ma 870/5-1 (Leibniz Award Ernst W. Mayr)

- Given two three-dimensional structures A and B, is A part of B? Where does it occur?
- Given two three-dimensional structures A and B, what is the largest part that they have in common? Or for polypeptide chains: what are the longest segments that are folded in the same way?

These questions can be generalized into the following, when thinking about structure databases:

1. (a) How to search all (exact/approximate) occurrences of a (sub)structure in a database?
   (b) How to find the database entries that share a largest common substructure with a given structure?
2. Are there frequently occurring substructures in the database and how do they look like?

We will show, that it is possible to answer these questions in an appropriate way for the case of a protein structure database, if the entries are preprocessed to build a structural index.

1.1 Protein Structure

Since the focus of application is rather on proteins than other molecules, we want to restrict the solution to this special kind of macro-molecules and take advantage of their linear composition from the residues.

Polypeptides are composed from amino acids that are chained together by so called peptide bonds. The pure information about the sequence of amino acids is called the primary structure, but it doesn't say anything about (three-dimensional) structure at all (except, that it is a chain with certain residues). The so called secondary structure elements are formed by regularities of the backbone conformations. The most prominent ones are the α-helices as well as the β-strands which form the β-sheets. The term tertiary structure denotes the actual three-dimensional conformation of one polypeptide chain. The way how one or more polypeptides assemble a protein is referred to as the quaternary structure.

Several connected secondary structure elements may build a more or less complex supersecondary structure or *(structural) motif*. Examples are the beta hairpin, the Greek key, the helix-turn-helix / helix-loop-helix motif (in particular the DNA- and the calcium-binding motif), the zinc finger, the β-α-β motif as well as the jelly roll (that is made of several Greek key motifs).

For further information on protein structure please refer to the book by Branden and Tooze [2].

1.1.1 Dihedral angles

For the description of the three-dimensional conformation of a polypeptide chain, the torsion angles ϕ, ψ and ω are used (see right part of figure 2). They are advantageous compared to absolute positions of the backbone atoms because of their invariance to translation and rotation of the molecule in the actual coordinate system. These so called dihedral angles are defined by

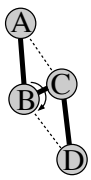
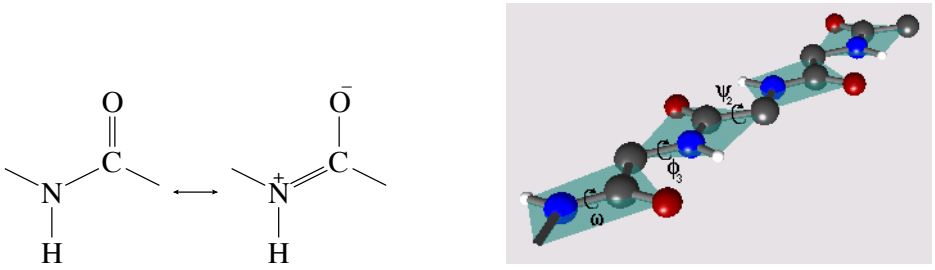


Figure 1: definition of a dihedral angle.



Note: the peptide chain starts in the upper right corner with the first C_α and elongates with the peptide bond $(CO)(NH)$ and the next $C_\alpha(CO)(NH)$-groups to the lower left corner.

Figure 2: The peptide bond and a polypeptide backbone with torsion angles.

the torsion of two bonds (AB and CD) around another bond (BC) or more formally by the angle between the planes ABC and BCD (see figure 1).

Due to resonance effects, the mentioned peptide bonds have a partial double bond character, which implicates a planar structure of the involved atoms ($C_{\alpha,i}$, C_i, O_i, N_{i+1}, H_{i+1} and $C_{\alpha,i+1}$). This fact reduces the degrees of freedom for the angle ω that measures the torsion around the peptide bond to only two values: $0°$ and $\pm 180°$. Since the trans configuration ($\pm 180°$) is almost always preferred to the cis conformation ($0°$), we don't take ω into consideration. So one can reduce the three-dimensional structure of the protein to its most significant parameters by describing the backbone as a sequence of the torsion angles near the C_α-atoms (ϕ_i and ψ_i).

Statistical analysis of the frequency of occurrence of such angle pairs has shown, that there are some combinations, that are found very often, and there are other combinations that only rarely or never occur. The diagram that plots these frequencies is called the Ramachandran plot [17]. It is a standard tool in the analysis and validation of protein structures.

2 Previous Work

Previous work has been done in the subject of searching substructures as well as in the topic of identifying frequent motifs. But most of the times the methods can only be applied to a pair or a small set of sequences and not to a whole large database. Another important point is that often only the sequence of amino acids is considered, only in rare cases the three-dimensional structure.

The work of Sagot, Viari, Pothier and Soldano [18] tries to find approximate repetitions in a set of protein structures by defining a similarity relation on the mesh partitions of the Ramachandran map. It extends an earlier algorithm by Soldano, Viari and Champesme that itself is an extension of an algorithm for finding exact repetitions by Karp, Miller and Rosenberg [10]. The method uses the concept of maximal cliques of the similarity relation and is one of the first that does not only apply dynamic programming on a pair of structure sequences.

Koch, Lengauer and Wanke [12, 11] tried to find maximal common substructures based on the secondary structure elements. They use a more graph-theoretical approach by modifying an algorithm by Bron and Kerbosch, which enumerates all maximal cliques of a graph. The aim is to find maximal common substructures in two or several graphs, where the nodes represent the secondary structure elements. This is done by reducing the problem to the MAXIMUM CLIQUE PROBLEM of the so-called product graph. By restricting the search to cliques that represent connected substructures the search space can be reduced.

Sagot et al. [19] give algorithms for finding subsequences that occur (with a certain error) in a given number of sequences of the sequence set. In this case, the methods are only applied to the

sequence of amino acids and not to the structure in space. The same holds for a paper by Marsan and Sagot [14], but here suffix trees are used in combination with the Hamming distance (instead of the Levenshtein/edit distance).

Escalier et al. [5] present an algorithm that computes recursively common three-dimensional substructures for two or several proteins. But this method can only be applied to small instances with about 30 atoms. Otherwise a two step approach with a Branch and Bound technique has to be applied.

Chew et al. [3] present an approach that is capable of computing common geometric substructures of only two molecules, but most interestingly the paper also addresses the problem of detecting common domains, i.e. larger substructures where proximity in space does not coincide with continuousness along the polypeptide chain. Another interesting point is the use of another backbone representation that is based on the virtual-bond vectors between consecutive α-carbon atoms.

Geometric Hashing, a method that came from the field of computer vision, was used to solve the structural alignment problem. For an application to the alignment of multiple structures and the detection of common motifs please refer to the paper by Nussinov and Wolfson [13]. This method is capable of searching for common domains that are not necessarily contiguous in one chain.

A closely related problem to the search for frequently occurring motifs is the k-COMMON SUBSTRING PROBLEM, where the aim is to find the length (and an occurrence) of a longest subsequence common to at least k of the K sequences (for all $k \in [2, K]$). Surprisingly, this problem can be solved in time $O(n)$ where n is the sum of the lengths of all sequences (see [8]). Please note, that this problem differs from our setting in the fact, that occurrences are counted only once per sequence, but we want to count every single occurrence (even multiple times in the same string).

A more detailed survey on different aspects of structure comparison and patterns is given by Eidhammer, Jonassen and Taylor [4].

3 The Polypeptide Angle Suffix Tree (PAST)

3.1 Suffix Trees

Suffix Trees are data structures, that efficiently solve the problem of searching all occurrences of a pattern P in a text T. After the suffix tree for T was constructed in a preprocessing step, the complexity of the search procedure does not longer depend on the length of T and is linear in the length m of P if one is only interested in the first occurrence. If all occurrences have to be found, the complexity increases to $O(m + k)$ where k denotes the number of occurrences of P in T.

For the efficient construction of suffix trees, there were three algorithms published. The first linear-time algorithm was proposed by Weiner [23] in 1973. A more space-efficient algorithm was provided in 1976 by McCreight [15]. Two decades later, Ukkonen [22] presented another linear-time construction, that can be used online, because it processes the text from the beginning to the end (unlike the algorithm of McCreight) and iteratively constructs the suffix trees for increasing lengths of the text. For an excellent review of these algorithms please refer to a paper by Giegerich and Kurtz [6].

Suffix trees can easily be extended to the case where more than one text (i.e. a set of sequences) has to be considered. These *Generalized Suffix Trees* can be used as index structures for databases. Further information on suffix trees and related structures can be found in Gusfields book [7].

3.2 Construction of the PAST

For the construction of our indexing data structure a variation of Ukkonens algorithm is used that is extended to generalized suffix trees. We go through all entries of the structure database, i.e. we inspect every entry of the PDB and compute for all polypeptide sequences the corresponding dihedral torsion angles from the peptides backbone atoms. These angles are coded into a new alphabet (e.g. represented by the characters with ASCII code from 1 to 24) by discretizing in intervals of $15°$. The character with ASCII code 0 is reserved for marking the end of a sequence to assert that each suffix is represented by a leaf in the suffix tree.

One of the advantages of this data structure is that an easy update of the index is possible if new entries are added to the database. Since an online algorithm is used it is not necessary to process all entries again. Only the angles for the new entries must be calculated and the respective sequences must be added to the suffix tree.

3.3 Searching Protein Structures

3.3.1 Exact Matching

Exact searching of a structure can easily be done by computing the dihedral torsion angles and coding them into characters analogously to the method described above (but of course the same interval coding as in the construction of the PAST has to be applied). Then the search process starts at the root node and always follows the path in the tree that is marked with the sequence of search characters. If the search gets stuck (i.e. there is no edge leaving the current node with the proper label), we know that the sequence is not a subsequence in an entry of the database. But otherwise we must distinguish the following cases:

1. If the search ends at a leaf of the tree, there would be exactly one occurrence in a suffix tree for only one text. But since we work with generalized suffix trees, we could have hits from more than one string at this position. We could either directly read the number of occurrences or inspect the whole list of occurrences, that is attached to the leaf node.
2. If the search ends at an inner node of the tree, we can only evaluate the number of occurrences in this node as well as the first occurrence position. If we want to know all positions, we have to traverse the whole remaining subtree. All leaves then correspond to an occurrence in the database.
3. If the search ends in the middle of an edge, we go to the next node and perform the operations analogously to the previous case.

The search procedure can easily be modified to compute the longest prefix of P that occurs in T (by simply searching until no further matching character is found). If the search follows the suffix link each time after the breakup, this can even be extended to compute the longest segment of pattern P, that occurs as subsequence somewhere in the database.

If we want to know what the longest common substructure of two or more database entries looks like, we simply traverse the whole tree and always remember the current depth. In each node we check whether the number of occurrences is greater than a given threshold and we keep the node with maximum depth.

If we apply the search method in its original form it has on the one hand a linear time complexity, but on the other hand we are facing the following problem: although the corresponding torsion angles of two structures are very close together, the derived codes might be separated by the discretization boundaries. Therefore these angles would be coded by different characters and the respective angle sequences would not match despite their geometrical similarity.

This leads to a more flexible way of searching but also to an increase of the computational complexity.

3.3.2 Approximate Matching

Approximate sequence matching is concerned with the question whether there exist similar, but not necessarily equal occurrences in the database. Often the similarity measure is defined by Hamming or Levenshtein (edit) distance. See e.g. a paper by Ukkonen [21], where a suffix tree is used to find approximate sequence matches. Another method is given by Hunt, Atkinson and Irving [9].

For our problem their distance or similarity measures are not suitable, because we only want to allow substitutions for codes of neighboring angle intervals. The pair of characters that code for the angles $5°$ and $175°$ has a much greater distance than the pairs that code for $5°$ and $-5°$ or $175°$ and $-175°$. Note that the characters $x_1, x_{|\Sigma|} \in \Sigma$ also code for very similar angles, because $-180°$ is the same as $180°$ and ASCII code 0 is the special chain termination symbol. However, choosing intervals of backbone angles already implements a certain error tolerance.

As already mentioned, the time complexity of the search is now much worse, because it might be exponential in the length of the search pattern. One possible solution to this problem would be to limit the number of mismatches. But our focus is on the application in practice where processing these queries is also very fast without this restriction (see experiments in the next section).

The search procedure can easily be extended to search for patterns that define different neighborhood ranges for each angle code, which is a very useful feature in practice, since some parts of the whole structure are more conserved or less flexible than others (e.g. helices). Another possibility to improve selectivity and sensitivity of the search is to compute an average structure of the hits. This can be used to repeat the search which yields more true positives with the same query parameters because local deviations in the torsion angles of the search pattern are reduced.

3.4 Identification of Structural Motifs

If we want to search for frequently occurring three-dimensional substructures (structural motifs), we simply have to define a lower bound for the length of the sequence and a lower bound for the number of occurrences. Then we traverse the whole tree and grab all nodes that match the conditions. This is very fast too because the complete suffix tree of the whole PDB can be held in the main memory.

In general the three-dimensional structure for proteins is more conserved than the sequence of amino acids. So we hope to find larger matching substructures with the conformation based approach described here than by using classical methods for sequence comparison.

A serious problem appears because of the many occurrences of the typical secondary structure elements (helices and strands). These are of course found in many variations and must be filtered out to find really interesting motifs.

4 Experiments and Results

We tested the application of our method to all entries of the PDB. For more than 40.000 chains from more than 20.000 PDB files roughly 18.5 million torsion angles of the backbone atoms were calculated, then coded with a certain interval size and finally added to the PAST.

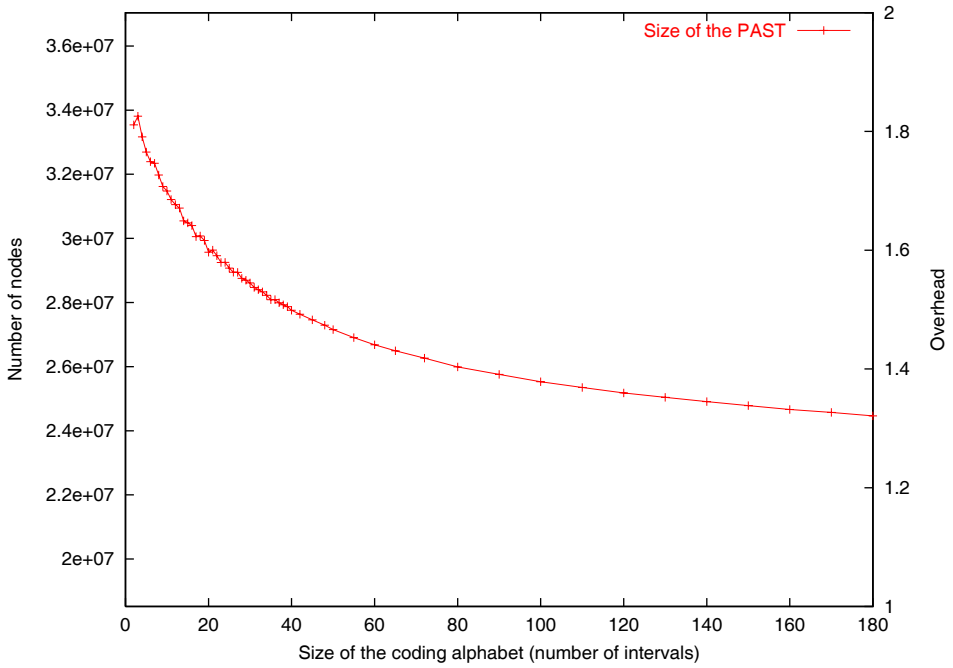


Figure 3: The size of the suffix tree as a function of the alphabet size.

Figure 3 shows the number of nodes and the overhead ratio in the suffix tree depending on the size of the coding alphabet (where overhead means the number of nodes divided by the number of angles or characters).

While parsing all entries of the PDB (unpacked more than 12 GB) and computing the torsion angles for the polypeptide chains takes about 1 hour on a 1 GHz PC, the computation of the generalized suffix tree itself takes less than 5 minutes(!). For this huge database (the file containing the angles information has a size of 150 MB) this provides also the possibility to calculate once the set of angles and then to compute the PAST for different alphabet sizes within a short time.

To test the capabilities of the PAST, we performed several searches and compared the results to PROSITE [20] pattern / profile matches and entries of SCOP [16] families. The results are shown in the tables 1 and 2. (The columns that are marked with S_i show the hits for a search with search range $\pm i$. The entries in the lower part of the table are the false positives.) Table 1 compares the matches of a search for zinc fingers of the C2H2 type to the respective groups in the PROSITE (PS00028 and PS50157) and SCOP (Classic zinc finger, C2H2) databases. The search pattern (i.e. coded angle sequence) was taken from PDB entry 1A1F (chain A, 46 angles from the residues 135–158). For the CCHC zinc finger type taken from PDB entry 1MFS (28 angles from the residues 15–29) a similar search was performed and compared to the PROSITE (PS50158) and SCOP (Retrovirus zinc finger-like domain) families.

Most of the PROSITE / SCOP group members are found. The missing entries have insertions compared to the pattern from the search sequence (in most cases between the two preserved histidine residues). This problem can be solved by allowing parts of flexible length in the search sequence of angle codes (like it is done for amino acid sequence patterns, e.g. in PROSITE). This could significantly increase the search time, but since traversing the whole PAST takes only a few minutes, this would be no real problem in practice.

Also a search with a whole α-chain of a hemoglobin molecule (chain A of PDB entry 1A3N)

was conducted (detailed data not shown because of the large number of hits). Due to the length of the search pattern a very high selectivity (no false positives even in the case of a range of ± 9) was observed.

The testing for large common substructures leads to the expected result that was mentioned above: the sequences are dominated by the typical secondary structure elements (mainly variations of α-helices and β-strands).

5 Conclusions

The method of discretizing the backbone angles and putting the respective codes into a generalized suffix tree has proven to be a very fast solution for answering structural queries to a huge database. The results of our experiments have also shown that the reduction of the complex three-dimensional structure of a protein to a simple sequence of torsion angle codes preserves enough structural information to yield very selective and sensitive query results. Even with short (angle) sequences and wide neighborhood ranges ($\pm 6 \widehat{=} 195°$) we found almost only true hits (see table 3). Also a rather sharp boundary (regarding the search range) was observed to the case where many false positives occur. The results also suggest that the discretization interval of $15°$ is yet more restrictive than necessary and could be increased to a much higher value without worsening the selectivity.

The search times of approximate matching show, as expected, an exponential runtime behavior. But even for a large search interval (neighborhood) the respective times for searching the whole PDB are in the order of several seconds which is orders of magnitudes faster than the computation time of other approaches.

6 Outlook

Besides the implementation of the above mentioned flexibility for varying lengths and acceptable deviations in the query sequence, a possible starting point to achieve improvements in sensitivity and selectivity is the adaption of the angle intervals, for instance in correspondence with dense and sparse regions of the Ramachandran plot.

Another approach could be to combine the search results for two or more structural motifs (that are non-consecutive along the chain) to find conserved domains.

Furthermore new possible structural motifs have to be verified using biological knowledge.

PDB code	Chain	Pos.	Sequence	PROSITE		SCOP	S_1	S_3	S_5
1a1f	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL						√
1a1g	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL						√
1a1h	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL						√
1a1i	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
1a1j	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
1a1k	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
1a1l	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
1aay	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
1ard		104	FVCEVCTRAFARQEHLKRHYRSHT	1	1	1			√
1are		104	FVCEVCTRAFARQEALKRHYRSHT	1	1	1			√
1arf		104	FVCEVCTRAFARQEYLKRHYRSHT	1	1	1			√
1bbo		2	YICEECGIRXKKPSMLKKHIRTHT	1	2	2			√
1f2i	G	1135	FQCRICMRNFSRSDHLTTHIRTHT	12	12	12			√
	H	2135	FQCRICMRNFSRSDHLTTHIRTHT					√	√
	I	3135	FQCRICMRNFSRSDHLTTHIRTHT					√	√
	J	4135	FQCRICMRNFSRSDHLTTHIRTHT				√	√	√
	K	5135	FQCRICMRNFSRSDHLTTHIRTHT					√	√
1fu9	A	9	KYCSTCDISFNYVKTYLAHKQFYC						√
1g2d	C	135	FQCRICMRNFSQHTGLNQHIRTHT	6	6			√	√
	C	163	FACDICGRKFATLHTRDRHTKIHL					√	√
	F	235	FQCRICMRNFSQHTGLNQHIRTHT					√	√
	F	263	FACDICGRKFATLHTRDRHTKIHL					√	√
1g2f	C	135	FQCRICMRNFSQQASLNAHIRTHT	6	6			√	√
	C	163	FACDICGRKFATLHTRTRHTKIHL					√	√
	F	235	FQCRICMRNFSQQASLNAHIRTHT					√	√
	F	263	FACDICGRKFATLHTRTRHTKIHL					√	√
1jk1	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL						√
1jk2	A	135	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3	√	√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
1jn7	A	9	KYCSTCDISFNYVKTYLAHKQFYH						√
1mey	C	5	YKCPECGKSFSQSSNLQKHQRTHT					√	√
	C	33	YKCPECGKSFSQSSDLQKHQRTHT					√	√
	C	61	YKCPECGKSFSRSDHLSRHQRTHQ					√	√
	F	5	YKCPECGKSFSQSSNLQKHQRTHT						√
	F	33	YKCPECGKSFSQSSDLQKHQRTHT					√	√
	F	61	YKCPECGKSFSRSDHLSRHQRTHQ					√	√
1p47	A	135	FQCRICMRNFSRSDHLTTHIRTHT					√	√
	A	163	FACDICGRKFARSDERKRHTKIHL					√	√
	B	135	FQCRICMRNFSRSDHLTTHIRTHT						√
	B	163	FACDICGRKFARSDERKRHTKIHL	1	1	1		√	√
1paa		132	YACGLCNRAFTRRDLLIRHAQKIH					√	√
1ubd	C	325	HVCAECGKAFVESSKLKRHQLVHT	4	4	4		√	√
1yui	A	34	ATCPICYAVIRQSRNLRRHLELRH	1		1			√
1yuj	A	34	ATCPICYAVIRQSRNLRRHLELRH	1		1			√
1zaa	C	35	FQCRICMRNFSRSDHLTTHIRTHT	3	3	3		√	√
	C	63	FACDICGRKFARSDERKRHTKIHL						√
2drp	A	111	YRCKVCSRVYTHISNFCRHYVTSH	4	4	4			√
	A	141	YPCPFCFKEFTRKDNMTAHVKIIH					√	√
	D	111	YRCKVCSRVYTHISNFCRHYVTSH						√
	D	141	YPCPFCFKEFTRKDNMTAHVKIIH						√
1nm2	A	228	YVSNKDGRAVASGTEVLDRLVGQV					√	√
1qu9	B	34	PVNPKTGEVPADVAAQARQSLDNV						√
	C	34	PVNPKTGEVPADVAAQARQSLDNV						√

Table 1: Hits from the search for zinc fingers of the C2H2 type.

PDB code	Chain	Pos.	Sequence	PROSITE	SCOP	S_2	S_5	S_7
1a1t	A	13	VKCFNCGKEGHIAKNCR		2			√
	A	34	KGCWKCGKEGHQMKDCT			√	√	√
1a6b	B	24	DQCAYCKEKGHWAKDCP	1	1			
1aaf		13	IKCFNCGKEGHIAKNCR	2	2		√	√
		34	RGCWKCGKEGHQMKDCT					
1bj6	A	13	VKCFNCGKEGHTARNCR		2		√	√
	A	34	KGCWKCGKEGHQMKDCT				√	√
1cl4	A	51	GLCPRCKRGKHWANECK		1			
1dsq	A	29	PVCFSCGKTGHIKRDCK	1	1			
1dsv	A	56	GLCPRCKKGYHWKSECK		1			
1esk	A	13	VKCFNCGKEGHTARNCR	2	2		√	√
	A	34	KGCWKCGKEGHQMKDCT					
1f6u	A	13	VKCFNCGKEGHIAKNCR	2	2		√	√
	A	34	KGCWKCGKEGHQMKDCT			√	√	√
1hvn	E	1	VKCFNCGKEGHIARNCR	1	1			
1hvo	E	1	VKCFNCGKEGHIARNCR	1	1			√
1mfs		13	VKCFNCGKEGHIAKNCR	2	2	√	√	√
		34	KGCWKCGKEGHQMKDCT					√
1nc8		7	IRCWNCGKEGHSARQCR	1	1		√	√
1ncp	C	22	KGCWKCGKEGHQMKDCT	2	2			√
	N	1	VKCFNCGKEGHTARNCR					√
2znf		1	VKCFNCGKEGHIARNCR	1	1			√
1e5u	I	9	IEIVGTGVKGKLPTVWL					√
1jt3	A	29	GTVDGTRDRSDQHIQLQ					√
1nik	L	50	DCGHRILLKARTKRLVQ					√
1waf	B	604	YLNEVCGTEGEAFVLYG					√

Table 2: Hits from the search for zinc fingers of the CCHC type.

Search range in 15°		±0	±1	±2	±3	±4	±5	±6	±7	±8
1a1f	True positives	1	11	26	40	55	61	62	64	65
	False positives				1	3	3	3	3	254
	Time [s]	< 1	< 1	1	2	3	4	5	8	12
1mfs	True positives	1	1	3	3	6	9	14	15	18
	False positives								4	99
	Time [s]	< 1	< 1	< 1	< 1	1	1	2	3	6
1a3n	True positives	1	17	87	120	132	135	138	144	146
	False positives									0
	Time [s]	< 1	< 1	1	2	3	4	6	8	12

Table 3: The number of true and false positives for the structure searches.

Acknowledgments

We appreciate the support and helpful suggestions by Jan Griebsch, Moritz Maaß, Volker Heun and Ernst W. Mayr. We also thank Thomas Lengauer for providing us with a paper that was not available in our library.

References

[1] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[2] C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, New York, second edition, 1999.

[3] L. P. Chew, D. Huttenlocher, K. Kedem, and J. Kleinberg. Fast detection of common geometric substructure in proteins. In *Proc. of the 3rd Ann. Int. Conf. on Research in Computational Molecular Biology (RECOMB)*, pages 104–113. ACM Press, 1999.

[4] I. Eidhammer, I. Jonassen, and W. R. Taylor. Structure comparison and structure patterns. Technical Report 174, Department of Informatics, University of Bergen, Bergen, Norway, July 1999.

[5] V. Escalier, J. Pothier, H. Soldano, and A. Viari. Pairwise and multiple identification of three-dimensional common substructures in proteins. *Journal of Computational Biology*, 5(1):41–56, 1998.

[6] R. Giegerich and S. Kurtz. From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, November 1997.

[7] D. Gusfield. *Algorithms on Strings, Trees, and Sequences – Computer Science and Computational Biology*. Cambridge University Press, 1997.

[8] L. C. K. Hui. Color set size problem with applications to string matching. In *Proc. of the 3rd Ann. Symp. on Combinatorial Pattern Matching (CPM'92)*, volume 644 of *Lecture Notes in Computer Science*, pages 230–243. Springer-Verlag, 1992.

[9] E. Hunt, M. P. Atkinson, and R. W. Irving. Database indexing for large DNA and protein sequence collections. *The VLDB Journal*, 11:256–271, 2002.

[10] R. M. Karp, R. E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *Proc. of the 4th Ann. ACM Symposium on Theory of Computing (STOC'72)*, pages 125–136. ACM Press, May 1972.

[11] I. Koch and T. Lengauer. Detection of distant structural similarities in a set of proteins using a fast graph-based method. In *Proc. of the 5th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB'97)*, pages 167–178. AAAI Press, June 1997.

[12] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *Journal of Computational Biology*, 3(2):289–306, 1996.

[13] N. Leibowitz, R. Nussinov, and H. J. Wolfson. MUSTA - a general, efficient, automated method for multiple structure alignment and detection of common motifs: Application to proteins. *Journal of Computational Biology*, 8(2):93–121, April 2001.

[14] L. Marsan and M.-F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *Journal of Computational Biology*, 7(3):345–362, August 2000.

[15] E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, April 1976.

[16] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.

[17] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7:95–99, 1963.

[18] M.-F. Sagot, A. Viari, J. Pothier, and H. Soldano. Finding flexible patterns in a text – an application to 3d molecular matching. *Computer Applications in the Biosciences*, 11(1):59–70, February 1995.

[19] M.-F. Sagot, A. Viari, and H. Soldano. Multiple sequence comparison - a peptide matching approach. *Theoretical Computer Science*, 180(1-2):115–137, June 1997.

[20] C. J. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. PROSITE: A documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics*, 3(3):265–274, September 2002.

[21] E. Ukkonen. Approximate string-matching over suffix trees. In *Proc. of the 4th Ann. Symp. on Combinatorial Pattern Matching (CPM'93)*, volume 684 of *Lecture Notes in Computer Science*, pages 228–242. Springer-Verlag, June 1993.

[22] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[23] P. Weiner. Linear pattern matching algorithms. In *Proc. 14th IEEE Ann. Symposium on Switching and Automata Theory*, pages 1–11. IEEE, 1973.