A Method for Fast Approximate Searching of Polypeptide Structures in the PDB

Hanjo Täubig* Arno Buchner† Jan Griebsch†

{taeubig|buchner|griebsch@in.tum.de}

Efficient Algorithms Group, Department of Computer Science,
Technische Universität München, D-80290 Munich, Germany
http://www14.informatik.tu-muenchen.de/personen/taeubig/ProSt/PAST.html

Abstract: The main contribution of this paper is a novel approach for fast searching in huge structural databases like the PDB. The data structure is based on an adaption of the generalized suffix tree and relies on an translation- and rotation-invariant representation of the protein backbone.

The method was evaluated by applying structural queries to the PDB and comparing the results to the established tool SPASM. Our experiments show that the new method reduces the query time by orders of magnitude while producing comparable results.

Keywords: protein structure, database index, dihedral angle, generalized suffix tree

1 Introduction

In recent years, major progress has been made in methods to determine protein structure experimentally. As a consequence, the Protein Data Bank (PDB) [BWF+00] today holds more than 26,000 structures and continues to grow by almost 90 structures per week. This offers unprecedented possibilities to join knowledge from different proteins, or to project knowledge to novel proteins - if similarities among proteins can be identified efficiently. Proven and efficient methods based on the (amino acid-) sequence of the proteins exist, yet an urgent interest in methods for structural comparison remains for several reasons: structural similarities provide better clues about the function of a protein than sequence similarities since the function of a protein is largely determined by its 3-dimensional shape. Furthermore, structure is known to be better preserved than sequence, and hence may reveal evolutionary or functional relationships even if a similarity among the sequences is no longer detectable. Finally, structural analysis may aid our understanding of the principles of protein folding and architecture.

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

However, existing methods are mostly based on exhaustive search and pairwise comparison of structures. As a consequence, their query time scales at least linearly with the number of structures to be searched. Due to the almost exponential growth in the number of structures in protein databases, this is considered unsatisfactory.

These facts underline the need for efficient methods that enable researchers to search protein databases for structurally similar proteins. Questions that should be addressed by such methods include: How can we search all (exact/approximate) occurrences of a (sub)structure in a database? How can we find the database entries that share a largest common substructure with a given structure? Are there frequently occurring substructures in the database, and what 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.

2 Related Work

Protein Searching In order to make pairwise comparisons practical for larger databases, classical methods like *DALI* [HS94], *VAST* [GMB96] or *CE* [SB98] use a two-phase approach. In the first ('filter') phase, a set of candidate proteins is generated. A variety of strategies exist for this phase including the alignment of secondary structure elements (SSEs), distance matrices, feature profiles, or combinations of the methods. In the following refinement phase the set of reported positives is constructed, for example by identifying matching C_α atoms and computing RMSD values between the query and all candidate structures. Due to the quickly increasing size of structure databases, newer approaches further trade off precision for speed. *TopScan* [Ma00] takes SSEs as the basic elements of the search. *ProtDex2* [AT04] speeds up the filtering phase by using indexing data structures to rapidly select structures that are similar to the query according to precomputed feature vectors.

The approaches described above share some common drawbacks. The candidate selection in the filter phase sacrifices accuracy for speed by relying on abstractions like distance matrices, SSE topology or feature vectors. Additionally heuristics are often employed to ignore unlikely database entries. Yet the methods of the refinement phase generally remain very expensive, restricting the filter phase to a small candidate set. As a consequence the user is left with the choice of accepting a very slow response to his query or greatly increasing the risk of pruning true positives in the filter phase. Furthermore, by relying on such features as SSEs, areas of high local similarity (*functional motifs*) may remain unidentified. This is most unfortunate since the functional motifs determine a protein's function to a large degree.

SPASM [Kl99] is one of the few programs to specialize in finding functional motifs. It is a two phase method, based on inter-molecule distance matrices, and is used as a reference to our approach in this work.

Motif Detection The detection of frequent structural motifs has been investigated using graph-theoretic abstractions (e.g. clique detection, see [SVPS95, KMR72, KLW96, KL97]), Geometric Hashing [LNW01] or a combination of both approaches [WKHK03].

Chew et al. [CHKK99] 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 alternative backbone representation that is based on the virtual-bond vectors between consecutive α-carbon atoms.

A detailed survey on different aspects of structure comparison and patterns is given by Eidhammer, Jonassen, and Taylor [EJT99].

3 The Polypeptide Angle Suffix Tree (PAST)

Protein Structure Since the focus of this application is on proteins, we take advantage of their linear composition from the residues. Polypeptides are composed of amino acids that are chained together by peptide bonds. The pure information about the sequence of amino acids is called the primary structure. The so-called secondary structure elements are formed by regularities of the backbone conformations.

Several connected secondary structure elements may build a more or less complex super-secondary structure or *(structural) motif*. One example is the Zinc finger motif which is used for testing the query capabilities of our approach. See Branden and Tooze [BT99] for further information on protein structure.

To describe the three-dimensional conformation of a polypeptide chain, the torsion angles φ, ψ, and ω are used. 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 the torsion of two bonds (AB and CD) around another bond (BC) or more formally by the angle between the planes defined by ABC and BCD.

Since ω most often equals $180°$, it is ignored in our approach in order to save memory. Thus the three-dimensional structure of the protein can be reduced 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). Also tested was a variant where the backbone is represented by the τ-angles, that measure the torsion of the virtual bonds between consecutive C_α atoms.

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 is constructed in a preprocessing step, the complexity of the search procedure no longer depends 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 $\mathcal{O}(m + k)$ where k denotes the number of occurrences of P in T.

Weiner (1973) and McCreight (1976) presented algorithms for the linear time construction of suffix trees. An algorithm proposed by Ukkonen [Uk95] is able to work online, i.e. to iteratively compute the suffix tree for a growing text. For an excellent review of these algorithms refer to Giegerich and Kurtz [GK97].

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 indexing structures for databases. Further information on suffix trees and related structures can be found in Gusfield [Gu97].

Construction of the PAST For the initial construction of our indexing data structure, an extension of Ukkonen's algorithm to generalized suffix trees is used. We iterate 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 encoded into an alphabet Σ (e.g. represented by the characters with ASCII code from 1 to $|\Sigma|$) by discretizing in intervals of size $360^\circ/|\Sigma|$.

An advantage 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 added to the suffix tree.

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 (clearly the same interval encoding as in the construction of the PAST must be applied). An exact structure search can be performed by simply using the regular suffix tree search method that starts at the root and looks for appropriate children until the search string ends or does not fit any child of the current node.

The search procedure can easily be modified to compute the longest prefix of P that occurs in T (by 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.

To determine what the longest common substructure of two or more database entries looks like, we simply traverse the whole tree while keeping track of 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.

The search method in its original form has 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 similar, the encoding might be different due to the discretization boundary. Therefore these angles would be encoded by different characters, and the respective angle sequences would not match despite their geometrical similarity. This brings us to a more flexible way of searching but also to an increase in the computational complexity.

Approximate Matching Approximate sequence matching is concerned with the question of whether there exist similar, but not necessarily equal, occurrences in the database. The respective similarity / distance measure is often defined by the Hamming or the edit distance, which has been investigated in the context of suffix trees by Ukkonen [Uk93] and by Hunt et al. [HAI02]. For our problem, their distance or similarity measures are not suitable, because we only want to allow substitutions for codes of neighboring angle intervals (i.e. interval codes that differ at most by δ from the search character). The pair of characters that encode for the angles 5° and 175° has a much greater distance than the pairs that encode for 5° and -5° or 175° and -175°. Note that the characters $x_1, x_{|\Sigma|} \in \Sigma$ represent very similar angles, because -180° is the same as 180°. However, choosing intervals of backbone angles already implements a certain error tolerance.

As previously mentioned, the worst case time complexity of the approximate search is much worse than the exact search complexity 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. However in practice, processing these queries is also very fast without this restriction (see experiments in section 4). This may be explained by the fact that the suffix tree is sparse compared to the sequence space, which means that the average branching factor of the nodes decreases with increasing depth within the suffix tree. For the case of an equidistant partition of the angle space and assuming equal frequencies of the coding characters, the average trie search complexity has been investigated by Maaß [Ma04]. Although these assumptions do not hold in our case, the given asymptotic run time of $\mathcal{O}(n^{\log_{|\Sigma|}(2\delta+1)})$ can be regarded as a rough estimate of the complexity of the approximate search procedure.

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 a consensus angle sequence of a set of related sequences. This can be used iteratively to improve the query structure which yields better results with the same query parameters because local deviations in the torsion angles of the search pattern are reduced. Due to the periodic nature of angles, it is not allowed to simply compute an average value for a set of angles. Therefore we estimate the consensus angle by calculating the center of gravity for the representatives of the angles on the unit circle. If this point is different from the point of origin, the respective angle of this vector can be regarded as a consensus or 'average' angle. The length of the vector is a measure for the conformity of the angle set.

4 Experiments and Results

We tested the application of our method to all entries of the PDB. For more than 50,000 chains from roughly 25,000 PDB files, more than 24 million torsion angles of the backbone atoms were calculated, then encoded with an interval size of 15°, and finally added to the

PAST. Parsing all files of the PDB (unpacked more than 15 GB) and computing the torsion angles for the polypeptide chains takes about 1 hour on a 1 GHz PC and results in a file containing a sequence of angles for each PDB entry. Please keep in mind that this step has to be performed only once. After this preprocessing step, the character encoding for a selected alphabet size and the computation of the generalized suffix tree itself takes less than 5 minutes.

We performed several searches and compared the results to PROSITE [SCH+02] pattern / profile matches and entries of SCOP [MBHC95] families. Due to space limitations, only a small example can be shown here that demonstrates the power of the method. The results are shown in the tables (columns marked with $\pm\delta$ show the hits for a search with maximum tolerance δ). Table 1 compares the matches of a search for zinc fingers of the CCHC type using different search tolerances. The search pattern (i.e. coded angle sequence) was taken from PDB entry 1MFS (chain A, 26 angles from the residues 15–28).

Most of the PROSITE / SCOP group members are found. The missing entries have insertions compared to the search pattern. 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.

A search with a whole α-chain of a hemoglobin molecule (chain A of PDB entry 1A3N) was also 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 tolerance of ± 9) was observed.

For the C2H2 zinc finger type taken from PDB entry 1A1F (chain A, 46 angles from the residues 135–158) a similar search was performed and compared to the results of a respective search by SPASM (C_α atoms only, AA independent). Compared to the results of PAST with a search tolerance of ± 4, the SPASM-based search missed many occurences while PAST did not find 1 hit identified by SPASM. Alternatively, SPASM can be used to search for fixed residues at certain positions (which leads to more hits), but this wouldn't be a sequence-independent search anymore. However, the search time of SPASM lies always between several minutes and roughly one hour while PAST completes all queries within seconds.

An overview of search results together with the running times for different settings of the neighborhood ranges is shown in Table 3.

5 Conclusions and Future Work

The method of discretizing the backbone angles and putting the respective character encoding into a generalized suffix tree has proven to be a very fast solution for answering structural queries on huge databases. Remarkably 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

PDB code	Chain	Pos.	Sequence	± 2	± 4	± 6
1mfs		13	VKCFNCGKEGHIAKNCR	√	√	√
1a1t	A	13	VKCFNCGKEGHIAKNCR		√	√
	A	34	KGCWKCGKEGHQMKDCT		√	√
1a6b	B	24	DQCAYCKEKGHWAKDCP			
1aaf		13	IKCFNCGKEGHIAKNCR		√	√
		34	RGCWKCGKEGHQMKDCT		√	√
1bj6	A	13	VKCFNCGKEGHTARNCR		√	√
	A	34	KGCWKCGKEGHQMKDCT		√	√
1cl4	A	51	GLCPRCKRGKHWANECK			
1dsq	A	29	PVCFSCGKTGHIKRDCK			√
1dsv	A	56	GLCPRCKKGYHWKSECK			
1esk	A	13	VKCFNCGKEGHTARNCR		√	√
	A	34	KGCWKCGKEGHQMKDCT			
1f6u	A	13	VKCFNCGKEGHIAKNCR		√	√
	A	34	KGCWKCGKEGHQMKDCT	√	√	√
1hvn	E	1	VKCFNCGKEGHIARNCR	√	√	√
1hvo	E	1	VKCFNCGKEGHIARNCR		√	√
1mfs		13	VKCFNCGKEGHIAKNCR	√	√	√
		34	KGCWKCGKEGHQMKDCT		√	√
1nc8		7	IRCWNCGKEGHSARQCR		√	√
1ncp	C	22	KGCWKCGKEGHQMKDCT			
	N	1	VKCFNCGKEGHTARNCR			
2znf		1	VKCFNCGKEGHIARNCR			√

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

with short (angle) sequences and wide neighborhood ranges ($\pm 6 \widehat{=} 195°$) almost no false positives were observed. (see table 3). Also, a rather sharp boundary (regarding the search range) was observed in the case where many false positives occur. A possible interpretation is that the discretization interval of $15°$ is more restrictive than necessary and could be increased to a much higher value without worsening the selectivity. However, choosing a relatively small interval size and accepting a bigger number of neighboring search intervals minimizes the effect of discretization errors.

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

Identification of New Frequent 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 also very fast because the complete suffix tree of the whole PDB can be held in the main memory.

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

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

PDB code	Chain	Pos.	PAST(± 4)	SPASM
1a1f	A	135	$\surd$	$\surd$
1a1f	A	135,163	$\surd$	$\surd$
1a1g	A	135,163	$\surd$	$\surd$
1a1h	A	135,163	$\surd$	$\surd$
1a1i	A	135,163	$\surd$	$\surd$
1a1j	A	135,163	$\surd$	$\surd$
1a1k	A	135,163	$\surd$	$\surd$
1a1l	A	135,163	$\surd$	$\surd$
1aay	A	135,163	$\surd$	$\surd$
1ard		104	$\surd$	$\surd$
1are		104	$\surd$	$\surd$
1bbo		2	$\surd$	
1f2i	G	1135	$\surd$	
	H	2135	$\surd$	
	I	3135	$\surd$	
	J	4135	$\surd$	
	K	5135	$\surd$	
	L	6135	$\surd$	
1fu9	A	9	$\surd$	
1fv5	A	9	$\surd$	
1g2d	C	135,163	$\surd$	$\surd$
	F	235,263	$\surd$	
1g2f	C	135,163	$\surd$	$\surd$
	F	235,263	$\surd$	
1jk1	A	135,163	$\surd$	$\surd$
1jk2	A	135,163	$\surd$	$\surd$
1jn7	A	9	$\surd$	
1kls	A	3	$\surd$	
1llm	C,D	204,260	$\surd$	
1m36	A	8	$\surd$	
1mey	C,F,G	5,33,61	$\surd$	
1p47	A,B	135,163	$\surd$	$\surd$
1paa		132	$\surd$	
1p7a	A	10	$\surd$	$\surd$
1ubd	C	325	$\surd$	$\surd$
1yui	A	34	$\surd$	
1yuj	A	34	$\surd$	
1zaa	C	35,63	$\surd$	
1znf				$\surd$
2adr	A	159	$\surd$	$\surd$
2drp	A,D	111,141	$\surd$	

Table 2: Search results for zinc fingers of the C2H2 type computed by PAST and SPASM. As false positives PAST found the following entries: 1ghs, 1nm2, 1mla and 1qu9.

Search tolerance		± 0	± 1	± 2	± 3	± 4	± 5	± 6	± 7
Interval size		15°	45°	75°	105°	135°	165°	195°	225°
1a1f	True pos.	1	13	32	46	61	68	68	68
	False pos.				4	7	7	9	9
	Time [s]	< 1	< 1	< 1	< 1	1	2	3	4
1mfs	True pos.	1	1	3	7	12	16	18	22
	False pos.							4	29
	Time [s]	< 1	< 1	< 1	< 1	1	1	2	3
1a3n	True pos.	1	17	87	120	132	135	138	144
	False pos.								0
	Time [s]	< 1	< 1	< 1	1	2	4	5	6

Table 3: The number of true and false positives and the query time for the structure searches with different tolerances.

Aside from 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.

Currently, PAST is evaluated as a fast search tool for structural similarities in the process of drug design in a research cooperation with ALTANA Pharma.

Acknowledgments

We appreciate the support by Prof. V. Heun, Prof. E. W. Mayr and Prof. St. Kramer. We also thank Moritz Maaß for valuable suggestions concerning the suffix tree implementation and analysis, as well as Prof. Th. Lengauer for providing us with useful information. Special thanks go to Stella Clarke for proof reading our concepts. Further we want to acknowledge the support by Jürgen Paal from ALTANA Pharma, the improved search functions by Uwe Römers, as well as the fruitful discussions with Sebastian Wernicke. We thank the anonymous reviewers for helpful considerations.

References

[AT04] Aung, Z. and Tan, K.-L.: Rapid 3D protein structure database searching using information retrieval techniques. *Bioinformatics*. 20(7):1045–1052. May 2004.

[BT99] Branden, C. and Tooze, J.: *Introduction to Protein Structure*. Garland Publishing. New York. second. 1999.

[BWF+00] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E.: The protein data bank. *Nucleic Acids Research*. 28(1):235–242. 2000.

[CHKK99] Chew, L. P., Huttenlocher, D., Kedem, K., and Kleinberg, J.: Fast detection of common geometric substructure in proteins. In: *Proc. 3rd Conf. on Research in Computational Molecular Biology (RECOMB'99)*. pp. 104–113. ACM Press. April 1999.

[EJT99] Eidhammer, I., Jonassen, I., and Taylor, W. R.: Structure comparison and structure patterns. Technical Report 174. Department of Informatics, University of Bergen. Bergen, Norway. July 1999.

[GK97] Giegerich, R. and Kurtz, S.: From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*. 19(3):331–353. November 1997.

[GMB96] Gibrat, J.-F., Madej, T., and Bryant, S. H.: Surprising similarities in structure comparison. *Current Opinion in Structural Biology*. 6(3):377–385. June 1996.

[Gu97] Gusfield, D.: *Algorithms on Strings, Trees, and Sequences – Computer Science and Computational Biology*. Cambridge University Press. 1997.

[HAI02] Hunt, E., Atkinson, M. P., and Irving, R. W.: Database indexing for large DNA and protein sequence collections. *The VLDB Journal*. 11:256–271. 2002.

[HS94] Holm, L. and Sander, C.: Searching protein structure databases has come of age. *Proteins: Structure, Function, and Genetics*. 19(3):165–173. July 1994.

[KL97] Koch, I. and Lengauer, T.: 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)*. pp. 167–178. AAAI Press. June 1997.

[Kl99] Kleywegt, G. J.: Recognition of spatial motifs in protein structures. *Journal of Molecular Biology*. 285(4):1887–1897. January 1999.

[KLW96] Koch, I., Lengauer, T., and Wanke, E.: An algorithm for finding maximal common subtopologies in a set of protein structures. *J. Comp. Biology*. 3(2):289–306. 1996.

[KMR72] Karp, R. M., Miller, R. E., and Rosenberg, A. L.: Rapid identification of repeated patterns in strings, trees and arrays. In: *Proc. 4th Symp. on Theory of Computing (STOC'72)*. pp. 125–136. ACM Press. May 1972.

[LNW01] Leibowitz, N., Nussinov, R., and Wolfson, H. J.: 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.

[Ma00] Martin, A. C.: The ups and downs of protein topology; rapid comparison of protein structure. *Protein Engineering*. 13(12):829–837. December 2000.

[Ma04] Maaß, M.: Average-case analysis of approximate trie search. In: *Proc. 15th Symp. on Combinatorial Pattern Matching (CPM'04)*. volume 3109 of *LNCS*. pp. 472–483. Springer. July 2004.

[MBHC95] Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C.: SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*. 247(4):536–540. 1995.

[SB98] Shindyalov, I. N. and Bourne, P. E.: Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*. 11(9):739–747. September 1998.

[SCH+02] Sigrist, C. J., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., Bairoch, A., and Bucher, P.: PROSITE: A documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics*. 3(3):265–274. September 2002.

[SVPS95] Sagot, M.-F., Viari, A., Pothier, J., and Soldano, H.: Finding flexible patterns in a text – an application to 3d molecular matching. *Computer Applications in the Biosciences*. 11(1):59–70. February 1995.

[Uk93] Ukkonen, E.: Approximate string-matching over suffix trees. In: *Proc. 4th Symp. on Combinatorial Pattern Matching (CPM'93)*. volume 684 of *LNCS*. pp. 228–242. Springer. June 1993.

[Uk95] Ukkonen, E.: On-line construction of suffix trees. *Algorithmica*. 14(3):249–260. 1995.

[WKHK03] Weskamp, N., Kuhn, D., Hüllermeier, E., and Klebe, G.: Efficient similarity search in protein structure databases: Improving clique-detection through clique hashing. In: *Proc. of the German Conference on Bioinformatics (GCB'03)*. volume I. pp. 179–184. October 2003.