BIOINFORMATICS APPLICATIONS NOTE Vol. 20 no. 9 2004, pages 1459–1461
DOI: 10.1093/bioinformatics/bth083

ParSeq: searching motifs with structural and biochemical properties

M. Schmollinger, I. Fischer, C. Nerz, S. Pinkenburg, F. Götz, M. Kaufmann, K.-J. Lange, R. Reuter, W. Rosenstiel and A. Zell*

Center for Bioinformatics, University of Tübingen, Sand 14, 72076 Tübingen, Germany

Received on October 7, 2003; revised and accepted on December 16, 2003
Advance Access publication February 12, 2004

ABSTRACT

Summary: Searches for variable motifs such as protein-binding sites or promoter regions are more complex than the search for casual motifs. For example, in amino acid sequences comparing motifs alone mostly proves to be insufficient to detect regions that represent proteins with a special function, because the function depends on biochemical properties of individual amino acids (such as polarity or hydrophobicity). Pure string matching programs are not able to find these motifs; hence, we developed ParSeq, a program that combines the search for motifs with certain structural properties, the verification of biochemical properties, an approximate search mechanism and a stepwise creation of the motif description by allowing to search on previously obtained results.

Availability: http://www-pr.informatik.uni-tuebingen.de/parseq

Contact: parseq@informatik.uni-tuebingen.de

Searching for regions with specific characteristics in DNA or amino acid sequences is a frequent problem in biology that has many facets. Beginning with the search for start and stop codons in DNA, the search becomes more complicated when looking for more variable motifs. These are, e.g. protein-binding sites or promoter regions, the latter often indicated by characteristic motifs in variable distance. In amino acid sequences, comparing motifs alone mostly proves to be insufficient in detecting regions that represent proteins with a special function, as the function depends, among others, on biochemical properties of individual amino acids (such as polarity, hydrophobicity or electric charge). Amino acids with similar biochemical properties can be exchanged without loss of function, but the character string is changed and will not be found by mere string-matching programs. In some cases, the functional sub-units of a protein can be described by defining length and order of parts of the sequence, which have predominantly certain defined chemical properties. For example, signal peptides of staphylococcus consist of three parts that can be approximately described in the following way: (1) positively charged region, average length of six amino acids, with usually three positive charges, (2) hydrophobic region, average length of 19 amino acids and (3) SPase-I cleavage site, consensus sequence: AXA. Such a defined sequence of motifs so far could only be searched for with purpose-built programs, e.g. search for lipoproteins (Babu and Sankaran, 2002) or signal peptids (Nielsen *et al.*, 1997, http://www.cbs.dtu.dk/services/SignalP/). In order to be able to search for user-defined motif sequences, regular expressions must be available to describe the motif sequence in a machine-readable way. In addition to that, the program has to provide the following options: (1) search for motifs that may have errors, (2) search for motifs at variable distances and (3) search for motifs of variable length with certain biochemical properties. ParSeq can be considered as a filter that allows finding interesting parts on a DNA or amino acid sequence. Afterwards, the hits can be analyzed with specific programs or in the laboratory (thus the program serves to narrowdown of the search area). With ParSeq, we are able to search for motif sequences on a whole bacterial genome in a few seconds. The quality of the results depends on how accurate the motif sequence can be described. A more accurate description results in less hits, but this approach bears the danger that regions that represent proteins with the desired function will not be considered. In contrast, a less accurate description may result in too many hits that are too extensive for further investigation. The program allows a stepwise approach to find the appropriate degree of accuracy because the query can be repeated using an increased amount of information. At the moment, scoring tables are used to estimate biochemical properties like those used in ProtScale (ProtScale, 2002/2003, http://us.expasy.org/cgi-bin/protscale.pl).

The structural properties of motifs can be formulated using regular expressions. Hence, we decided to choose the regular expression library as the main method for searching motifs. The main task is the integration of tests of biochemical constraints for amino acid sequences and the search for approximate patterns for amino acid and DNA sequences (Edit and Hamming distance). In the first step,

*To whom correspondence should be addressed.

a language was designed with which it is possible to incorporate queries for these two aspects. In the second step, a search algorithm that combines calls to regular expression libraries and verification routines for biochemical constraints had to be built. Concerning the query language, it is allowed to use the powerful possibilities of regular expressions as defined by the existing libraries. In general, a query is nothing else than a chain of valid regular expressions. The chain links are separated by the character '@'. The following example shows a chain of three regular expressions: <regex>@<regex>@<regex>. In order to assign a test of a biochemical constraint to a regular expression, a semicolon separated list of calls to functions is added with a leading character '/' before the separation sign '@' as illustrated in the following example

$$\text{<regex>@<regex>/fct}_1(\text{arg}_1, .., \text{arg}_n); .. ;$$
$$\text{fct}_m(\text{arg}_1, .., \text{arg}_k)\text{@<regex>}.$$

Each function corresponds to the test of one biochemical constraint. In general, for each biochemical constraint there is one table in which a certain value for each amino acid is given. Within the function, a test is implemented that works on the values of a particular table and its result is either true or false. For example, a rational function is to test whether the mean value of a certain constraint is larger (equal, smaller) than a given value. The search for approximate patterns can be regarded as a special case of a constraint. We use the same syntax, but the functions can only be applied to fix query strings and not to general regular expressions. Both functions, the Edit (ed) as well as the Hamming distance (hd), take the number of allowed errors as their argument. An example query, with a maximal error of 1, might look like this: AAGGT/ed(1)@.

After we defined the query language, the search can be sketched as follows. First, the query is divided into parts that are pure regular expressions and into the constraints. Second, we start the search for candidates by using the pure regular expression. In the final step, the candidates are tested if they are valid concerning the biochemical constraints. Queries containing variable ranges with biochemical constraints are more complex, because, by default, regular expression libraries return the largest match. This match might not fulfill the biochemical requirements, but there might be others within the match that fulfill the constraints. Hence, we have to continue the search at this position until a match fulfills the requirements or no further match exists. Concerning the approximate search, the procedure is different. Roughly speaking, before the search starts the fixed string is translated into a more complex regular expression that covers all possibilities of its appearance. This is mainly done by creating a large expression that combines all possibilities by the logical OR operation.

Additionally, the search is also able to work on results achieved in earlier searches. This is a very important concept with respect to the running time and the interactive use of the program. The usual batch approach to motif search assumes that the query motif is known. Hence, the task of the search engine is simply to find its occurrences in the sequences. This is a very idealized view, which seldom describes the reality. More often, the motif itself is also not completely known, and the researcher has only a more or less rough idea what he is looking for. In such a constellation, the search is a lengthy trial-and-error process, in which the biologist starts with a simple and very general query and gradually narrows the possibilities as he or she approaches the result. Sometimes the refinement of the query leads to a wrong direction, or to no matches at all. In that case, it is desirable to be able to recall the previous query and modify it before repeating the search. On the other hand, repeating the whole search each time when the query is modified is unnecessary and, for complex queries, can be a real annoyance and can take minutes, or even hours. But if the new search is simply a refinement of an older one, it can be limited to the results of the old search. Our search engine supports this kind of incremental searching by keeping track of previous searches and allowing the user to take any of them as the basis for further searches. The history of searches is represented graphically as a tree. By default, a search starts from the root node of the tree, which represents a search on the whole sequence. Results of the search are the child nodes in the tree. If the user selects a node for further searching, the search is limited to the results associated with the node and is performed around the positions at which a hit was encountered. The engine compares the new, refined query with the previous one and decides in which direction to search further. If the new query differs from the old one only by a prefix, the search is performed only to the left of the hits. The same holds for the right direction. The results are assigned to a new node, one hierarchy level below. This interactive approach is not only intuitive, but can also be much faster than the straightforward search for complex motifs in the whole database. Finding fixed patterns in sequences, or even with few mismatches, can be performed very fast. Checking the biochemical properties of the matches is a much harder problem. Therefore, it makes sense to first narrow the search scope to sequences and positions in them containing easy-to-find 'anchor' patterns, and only then perform the expensive biochemical search. In our experiments, the sum of times needed for each step of the iterative search was up to an order of magnitude less than the duration of the same search when performed as a single, complex query.

ParSeq is written in Java (version 1.4+) and can be started using Java-Web-Start technology (SUN Microsystems, 2002, http://java.sun.com/products/javawebstart/). From up to this version, Java includes a regular expression library (Nourie and McCloskey, 2001, http://developer.java.sun.com/developer/technicalArticles/releases/1.4regex/).

ACKNOWLEDGEMENT

This work is supported by the 'Landesforschungsschwerpunktprogramm' of Baden-Württemberg, Germany.

REFERENCES

Babu,M.M. and Sankaran,K. (2002) DOLOP—database of bacterial lipoproteins. *Bioinformatics*, **18**, 641–643.

Nielsen,H., Engelbrecht,J., Brunak,S. and von Heijne,G. (1997) A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Int. J. Neural Syst.*, **8**, 581–599.

Nourie,D. and McCloskey,M. (2001) Regular Expressions and the Java Programming language.

ProtScale (2002/2003) The ExPASy (Expert Protein Analysis System) proteomics server of the Swiss Institute of Bioinformatics.

SUN Microsystems (2002) Java-Web-Start Technology.