Diplomarbeit

Heuristic database mapping methods for comparative proteomics

Frank Hager

March 15, 2007

Prüfer: **Prof. Bernhard Haubold**
FH Weihenstephan, Fachbereich Biotechnologie und Bioinformatik
Prof. Dimitrij Frishman
TU München, Lehrstuhl für genomorientierte Bioinformatik
Betreuer: **Thorsten Schmidt**
TU München, Lehrstuhl für genomorientierte Bioinformatik

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass die vorliegende Arbeit von mir selbst und ohne fremde Hilfe verfasst und noch nicht anderweitig für Prüfungszwecke vorgelegt wurde.
Es wurden keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt. Wörtliche und sinngemäße Zitate sind als solche gekennzeichnet.

Freising, den

Frank Hager

Contents

1 Summary **7**

2 Introduction **8**

3 Methods **15**

3.1 PROMPT Fast Mapping 15

3.1.1 Workflow 15

3.1.2 Heuristic methods 17

3.1.2.1 Heuristic A (more accurate) 17

3.1.2.2 Heuristic B (faster) 17

3.1.3 Evaluation of PROMPT - Fast Mapping 18

3.1.3.1 Test data 18

3.1.3.2 BLAST 20

3.1.3.3 PROMPT Fast Mapping 22

3.2 Implementation 22

4 Results **26**

4.1 Runtimes 26

4.1.1 BLAST 26

4.1.2 PROMPT Fast Mapping 27

4.2 Seed lookup 31

4.3 Accuracy 34

4.4 Availability 34

5 Discussion **35**

6 Conclusion **38**

7 Appendix **39**

A The PROMPT framework **39**

B The SIMAP database **47**

C Length distributions **48**

D Database download links **51**

List of Figures

1 Growth of SwissProt, PDB and GenBank database 9
2 Work flow of the PROMPT Fast Mapping method 16
3 Fast Mapping wizard dialog 24
4 Runtimes of BLAST runs 27
5 Runtimes of BLAST runs compared to those of PROMPT Fast Mapping runs 28
6 Runtimes of BLAST runs compared to those of PROMPT Fast Mapping runs using a mutated query sequence set 30
7 Histogram of sequence identities of seed and query sequence 31
8 Alignment seed/query 33
9 Screenshot of the PROMPT GUI 39
10 Prompt input interface diagram 42
11 Custom XML format of PROMPT. 43
12 Overview of the PROMPT main interface structure 45
13 Length distribution of database files downloaded from the respective download sites. 51

List of Tables

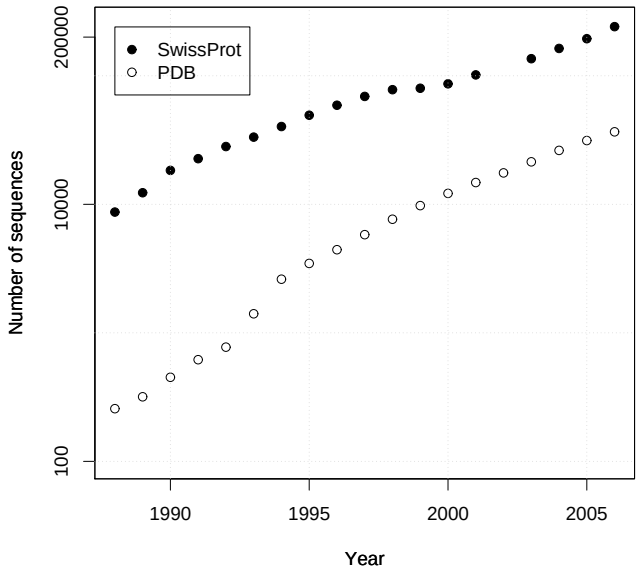
1 Public databases that are included in SIMAP 13
2 Distribution of sequence lengths in SwissProt 19
3 Subject databases used for evaluation 20
4 SIMAP databases 21
5 Runtimes of PROMPT mapping runs 29

1 Summary

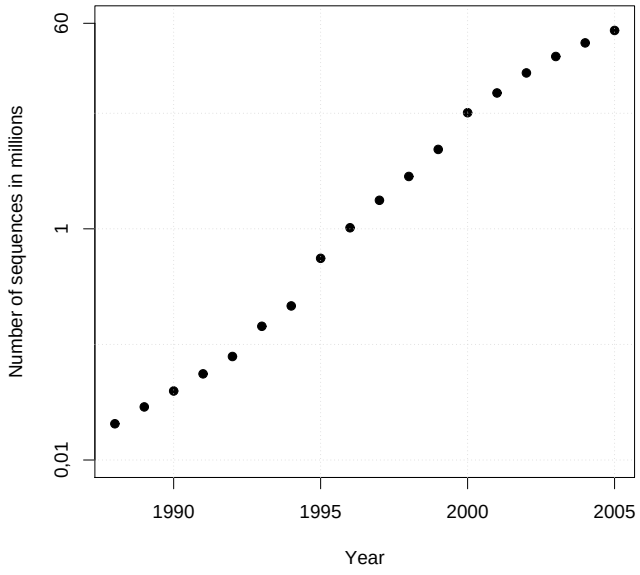
Finding sequence similarities is a fundamental task in bioinformatics. Even with well matured tools like the Basic Local Alignment Search Tool (BLAST), searching in large databases is a time consuming part of an analysis. A database of pre calculated similarities is useful, presuming the data is up to date. Here, a new method is developed to speed up the process of similarity searches up to 28 times or more compared to an equivalent search with BLAST. A database containing pre calculated sequence alignments (SIMAP) is used and two heuristic methods are implemented to retrieve hits for sequences that do not have pre calculated alignments in the database. The usage of the SIMAP database and heuristic methods makes this tool a fast and accurate method for similarity searches.

2 Introduction

Similarity searches among proteins are a fundamental and recurring task in bioinformatic disciplines like annotation and comparative proteomics. Sequence homology is used to infer knowledge from known to unknown sequences (e.g. Wilson *et al.* (2000)). The foundation for this procedure is the rule that proteins with similar sequences have a similar structures and thus similar functions. This is an established assumption in comparative proteomics and is taken as a basis for nearly all of those studies. During the past 20 years, biological databases have grown rapidly due to in refinements in sequencing techniques and new high-throughput sequencing methods, e.g. Pyrosequencing (Elahi and Ronaghi, 2004). With automation cost-effectiveness grows. Compared to the per base price at the beginning of the Human Genome Project, the price now is reduced 100-fold (Collins *et al.*, 2003). It fell from US $ 10 in 1990 per finished base to an estimated US $ 0.10 in 2005. There are many well established public databases like SwissProt (Wu *et al.*, 2006), GenBank or PDB (Berman *et al.*, 2000) which are constantly updated and grow steadily. The SwissProt database shows a yearly growth of around 20 percent for the last 4 years. Figure 1 shows this growth for some databases over the past 18 years. Figure 1(a) shows the growth in the number of public available amino acid sequences. For SwissProt, theses are only manually curated sequences. The growth in the number of nucleotide sequences in figure 1(b) for GenBank is even faster. Regarding this fast growth in the amount of sequence data, the need for efficient search tools becomes more and more important. Searches for corresponding sequences in different databases are sequences based. Mapping proteins from one database to another using its identifier as proposed by Draghici *et al.* (2006) does not accomplish this task properly. Identifiers may change through updates in databases or a new set of identifiers is introduced, which can lead to erroneous mapping. The sequences



(a) PDB/SwissProt



(b) GenBank

Figure 1 Growth of the databases SwissProt/PDB and GenBank from 1988 to 2006. Figure a shows the growth of protein sequences. Figure b the growth of nucleotide sequences. Source: SwissProt/PDB/GenBank statistic information

based approach is far more reliable, as the alignment produced by algorithms like Smith-Waterman (Smith and Waterman, 1981) or heuristic programs like FASTA (Pearson, 1990) or BLAST (Altschul *et al.*, 1990) always allow an evaluation of the mapping, provided scores or expectation values can be pulled up to decide whether a mapping is appropriate for a specific application. Execution times of exhaustive searches for homologous sequences in databases grow with the size of the respective database. Optimal sequence alignments based on the Smith-Waterman algorithm is usually too computationally expensive and even with well matured and optimized heuristic search algorithms like the Basic Local Alignment Search Tool, BLAST or FASTA, these searches can reach long runtimes. For example a BLAST search of a query set with 1000 sequences against the GenBank (Benson *et al.*, 2005) database with approximately 4 million sequences and 1 billion residues takes around 10 hours of search time . There are proposals for new tools to speed up similarity searches (Kent, 2002; Stephens *et al.*, 2003). But these methods may have prerequisites like a specific database management system for ODM (Oracle Data Mining) BLAST or need some pre calculation like *formatdb* for BLAST. Research institutes cannot afford to update their hardware equipment as fast as the sequence databases grow to keep search times short so new software tools are very important.

At some point, an often repeated BLAST or FASTA search can be replaces by a database of pre calculated sequence alignments. Rattei *et al.* (2006) proposed and implemented a database of pre calculated protein homologies based on the FASTA heuristic, called SIMAP, the similarity matrix of proteins. A precondition to such a database is the timeliness of the assembled data. SIMAP updates most of its assembled databases on a weekly basis. The problem with databases like SIMAP is the fact that sequences that are not contained in the database will not produce a hit and so is not suitable for every kind of bioinformatics analysis. Mapping sequences

from one database to another to find correspondences may succeed because of the vast amount of public databases included in SIMAP, but mapping newly sequenced data may return no hit. As of November 2006, SIMAP contains over 6 million sequences combined of various publicly available sources like UniProt or Genbank. The SIMAP database holds the alignments for every sequence against all other sequences in the database, yielding $\sim$ 10 billion single hits. For every sequence added to this database, new alignments are calculated against all other sequences using the FASTA program. To accomplish this enormous computational task, these calculations are performed using a grid system and in the near future also with a distributed computing project BOINC (Berkeley Open Infrastructure for Network Computing) (Anderson, 2004) providing enormous computing power by voluntary contributors. With databases like SIMAP, homology searches are reduced to database queries which means an enormous speed gain compared to a BLAST search.

In this work, a new method to find sequence homologies is developed. This method makes heavy use of a database of pre calculated homologies (SIMAP). Specifically heuristic methods are implemented to find a hit for queries whose sequence is not yet included in the database. A fast similarity search is used to determine a similar sequence to a sequence that is not included. This similarity search works with a reduced amino acid alphabet that combines amino acids with similar chemical characteristics. Sequences are divided into shorter fragments using a sliding window approach. These fragments are then indexed in a suffix array. For every fragment, an entry will contain all references to sequences that contain this fragment. To find a similar sequence, the number of fragments of a sequence that correspond to the query sequence is counted. If this number is greater than a threshold, the sequence is returned. Assuming this sequence will find a similar hit like the original query sequence, homologous sequences will be searched in the database.

Different heuristic methods to determine a hit are available and can be applied to the resulting sequence to retrieve a hit in a more accurate or faster manner. This method, called PROMPT Fast Mapping is implemented as part of the PROMPT (Schmidt and Frishman, 2006) framework. PROMPT (Protein Mapping And Comparison Tool) is a system to analyse, map and compare protein sets. For a detailed description of the PROMPT framework, see appendix A.

Application to comparative proteomics

Comparative proteomics provides insights into differences between protein sets or whole organisms' proteomes thus revealing interesting facts or complex connections of these sets or organisms. The foundation of comparative analyses is the availability of annotated information. Current knowledge about proteins is stored in various databases around the world, each addressing different parts of biological research. They are developed by different search groups, the data may be redundant and as a result the standardization of data may be a difficult task. PROMPT Fast Mapping addresses this problem. SIMAP allows the mapping of sequences against various public available databases (table 1).

To access data from the databases mentioned in table 1, the user can map his own set of sequences to the respective database to retrieve corresponding identifiers. With the identifiers, all information annotated to this sequence can now be accessed in the respective database. A rich resource for annotation data is the PEDANT database (Frishman *et al.*, 2003; Riley *et al.*, 2006; Frishman and Mewes, 1997) which is also included in SIMAP.

Available databases		
RefSeq	UniProt/TrEMBL	UniProt/SwissProt
NCBI COGs	PDB	GenBank
UniRef50	UniRef90	UniRef100
PEDANT 2	PEDANT 3	

Table 1 Public databases that are included in SIMAP. Most databases are updated on a weekly basis providing up-to-date data. These databases can be used for mapping. PROMPT Fast mapping also allows mapping against genome databases included in PEDANT

The PEDANT database The PEDANT genome database provides exhaustive automatic annotation of publicly available genomes. Established in 1996, its mission is to fill the gap between the few by human experts manually curated sequences and the vast amount of sequences produced by various genome sequencing projects. Currently it contains annotations for 503 completely finished and unfinished genomes. The annotation is performed by the PEDANT software suite which includes established bioinformatics methods. Functional characterization of protein sequences includes for example automatic assignment of functional categories (Ruepp *et al.*, 2004) and clusters of orthologous groups Tatusov *et al.* (2003). Structural assignment is based, among other methods on similarity searches against the Protein Data Bank (PDB) and secondary structure prediction. Additionally, PEDANT contains manually curated genomes like *Saccharomyces cerevisiae*, *Thermoplasma acidophilum* or *Arabidopsis thaliana*. The PEDANT database is available via a web based interface at http://pedant.gsf.de as well as web service for proprietary applications. The integration of PEDANT into the PROMPT framework offers the user access to the rich comparison and analyse methods offered by

the PROMPT framework. PROMPT allows the user to import sequences of whole genomes as well as specific sequences of a genome. Additionally, a combination of sequence identifier and organism identifier can be used to retrieve annotated information of the respective sequences using PEDANT. Due to the possibility to map sequences not only to well known databases like UniProt or PDB but also the PEDANT database it is possible to import sequences previously mapped against the PEDANT database into PROMPT in order to access the vast amount of annotated information of PEDANT.

3 Methods

3.1 PROMPT Fast Mapping

A method to query arbitrary sequence homologies from a database of pre calculated homologies was implemented. This includes the querying of the pre calculated homologies as they are included in SIMAP as well as homologies of sequences that are not included in the database making it suitable for arbitrary sequences. To find homologies of sequences not included, two heuristic methods to get a hit were implemented. A faster heuristic and a more accurate heuristic are available.

3.1.1 Workflow

The method takes as input an arbitrary amino acid sequence that is refered to as query sequence. For this query sequence the best homologous hit in a user defined search space will be returned. The search space can be a single genome database like *Escherichia coli K12* or a combination of multiple single databases e.g. all bacterial genomes.

The first step is to check whether a pre calculated search result is available in the SIMAP database (see top left in figure 2). To determine whether the query sequence is included in SIMAP a MD5 checksum (Rivest, 1992) of the sequence is calculated. Sequences are indexed by their MD5 value in SIMAP. With this checksum the SIMAP server is queried to return the best homologous protein hit along with hit statistics. In all cases in which no pre calculated results are available in SIMAP, the method provides two heuristics to estimate the most similar sequence hit utilizing the existing similarity matrix of SIMAP. With this most similar sequence and the implemented heuristic methods, a result hit will be determined. All communications with SIMAP are accomplished using Web Services (Weerawarana

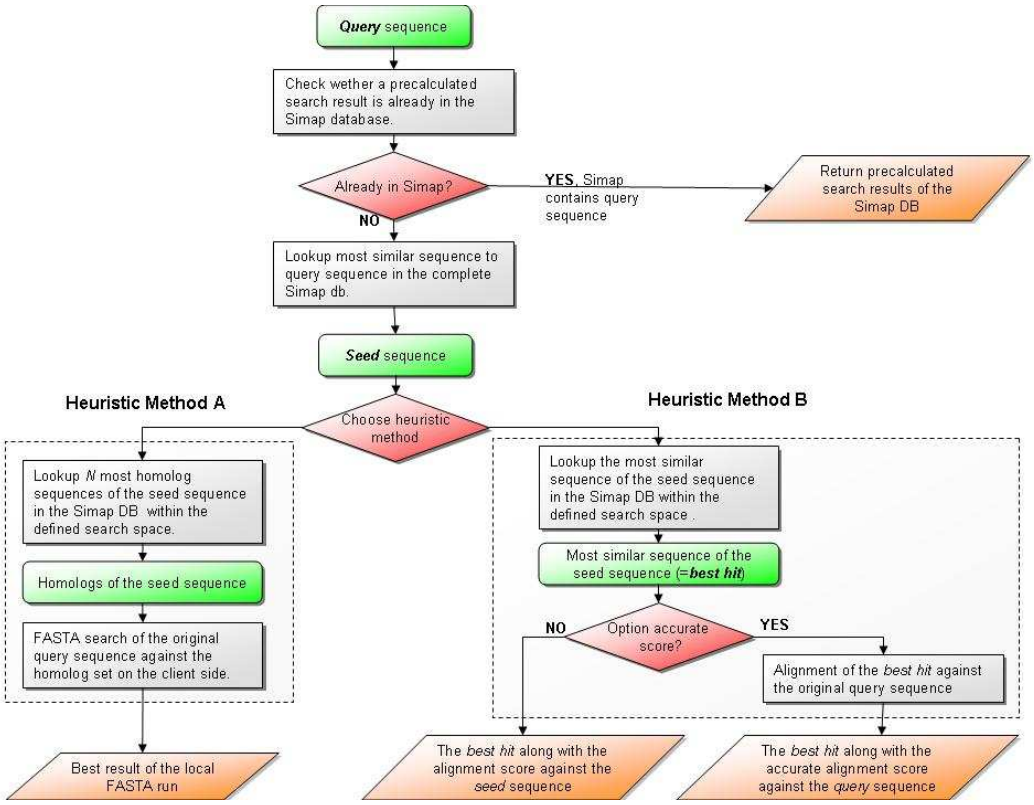


Figure 2 Work flow of the PROMPT Fast Mapping method. The first step is to check whether a pre calculated result is available in the SIMAP database; this result would be returned. If no pre calculated results are available in SIMAP yet, the most similar sequence in the defined search space will be soughed and upon this sequence and the heuristic method chosen, the estimated best hit will be returned

et al., 2005).

3.1.2 Heuristic methods

As starting point of both heuristics the sequence in SIMAP is looked up that has the highest degree of sequence similarity by a server side search. The assumption is that this seed sequence, if similar enough to the query sequence, will have a similar hit in the SIMAP database like the query sequence. From this point, the result will be determined by one of the heuristic methods.

3.1.2.1 Heuristic A (more accurate)

This is the more accurate heuristic. Using the seed sequence, determined by the most similar sequence included in SIMAP to the original query sequence, the 50 sequences most homologous to the seed sequence are looked up. The result constitutes a new sequence set. To retrieve the most homologous sequence to the query sequence, a FASTA search is performed using the query sequence and the new sequence set. The selection of the 50 best hits means a reduction of the search space from over 6 million sequences currently included in SIMAP to only 50 sequences. This reduced search space is 120000 times smaller than SIMAP's whole sequence space. The best hit of this search will be returned as result. The retrieval of 50 best homologs, the construction of the new sequence set and the FASTA search lead to a longer runtime of this heuristic method, but also generate the most accurate results.

3.1.2.2 Heuristic B (faster)

In high throughput environments, using the first heuristic and querying many sequences that have no hit in the SIMAP database can result in slightly longer runtimes. So the second heuristic produces fast results at the cost of accuracy. Starting from the seed sequence, the best hit in the SIMAP database to the seed sequence is returned. Additionally, the user can request a new score, calculated by aligning

the best hit of the seed sequence to the query sequence using the Smith-Waterman algorithm (Smith and Waterman, 1981).

3.1.3 Evaluation of PROMPT - Fast Mapping

To evaluate the performance of this mapping method, the CPU time of a blast search is compared to the CPU time of this method in various test settings. The accuracy is evaluated by comparing the results of this method to those returned by a BLAST search.

3.1.3.1 Test data

To assess the performance and accuracy of the new method, especially the heuristic methods, sequences have to be used that are not included in SIMAP. Sequences that are included will not make use of our heuristics as they will only be queried and SIMAP will return the best homologous hit. Multiple test datasets for our evaluation were created. Random sets which consist of sequences that are created totally at random and mutated sequence sets that consist of sequences with some single point mutations. Additionally, a sequence set with sequences that are all contained in SIMAP is created.

Random sequence set

1000 sequences were created by random. This randomization of amino acid sequences depends upon the amino acid distribution of the SwissProt database. The actual amino acid distribution is published with every new SwissProt release. For this randomization the distributions of the 51.1 release of UniProtKB/SwissProt was used. The length distribution of the sequences is also taken from this SwissProt release.

Property	Value
mean	367.06
min	2
max	34350
std	344.23

Table 2 Length distribution of sequence length in SwissProt release 51.1 (241365 sequences)

Mutated sequence set

1000 sequences were randomly extracted from the UniProt/TrEMBL database file. They were mutated by randomly selecting 2 positions on the sequence and switching the respective amino acid with the acid on the other position. The sizes of the sequences were taken into account by mutating an average of 10% of each sequence. As one swap results in the mutation of two amino acid positions the number of swaps is 0.05*sequence length.

Sequence set contained in SIMAP

From the UniProt/TrEMBL database file, 1000 sequences were randomly chosen and extracted into a new sequence file. These sequences will all have a hit in SIMAP because the UniProt/TrEMBL database is included in SIMAP and homologies are calculated for all those sequences. This set of sequences was also used to generate the mutated sequence set as described above.

Subject databases

For the BLAST run, the query sequence sets were searched against databases of different size. The databases were downloaded from the respective database sites.

Links are provided in appendix D. Figure 3 shows the sequence and amino acid count for the respective databases. Length distribution histograms for every database can be found in appendix C.

Database	Number of sequences	Number of amino acids
Protein Data Bank	25 318	18 535 516
UniProt/SwissProt	230 133	84 471 903
UniRef50	1 292 284	452 580 312
UniRef90	2 475 002	888 282 121
UniProt/TrEMBL	3 051 654	987 652 774
UniRef100	3 867 872	1 399 869 765
GenBank	4 210 908	1 179 840 948

Table 3 Subject databases used for evaluation. Beside the number of sequences in the database, the number of amino acids is stated. The relevant number determining the runtime of a BLAST run is the number of the residues in a dataset

SIMAP databases

To compare the runtimes of the BLAST search and the search using PROMPT Fast Mapping, the same databases that were downloaded were used. The databases that were used for evaluation are listed in table 4.

These number of sequences and amino acids does not correspond exactly to the downloaded databases. The sizes differ slightly but that does not affect the evaluation of PROMPT Fast Mapping so that the results can be compared.

3.1.3.2 BLAST

As a reference time for a databases search, the runtime of a BLAST search was taken. Default parameters were used in blastall with the exception of option -b,

Database	Number of sequences	Number of amino acids
Protein Data Bank	27 369	21 158 580
UniProt/SwissProt	217 260	86 495 188
UniRef50	1 285 072	444 962 284
UniRef90	2 460 593	876 559 253
UniProt/TrEMBL	3 003 420	1 068 499 857
UniRef100	3 843 153	1 390 142 151
GenBank	4 210 908	1 179 840 948

Table 4 Databases and sizes currently (release Nov 16, 2006) included in SIMAP

specifying the number of sequences for which an alignment should be shown in the result. Only the best hit in the database is needed, so the *-b* option can be changed to 1 instead of default 250. In this evaluation databases of different size to search against were used. The runtime of the BLAST algorithm depends on the size of the database you search against and the size of the query sequence set. The total amino acid count is the relevant size of a database. Two datasets with the same number of amino acids but different number of sequences need the same time to execute. For example a dataset with 100 sequences of length 1000 and a dataset with 50 sequences with length 2000 will run in the same time. Both contain 100 000 amino acids but the first set has twice the number of sequences. The speed gain of searching against large database is to be assessed, so different sizes of subject sequence sets were used. For this evaluation seven different databases with different number of sequences included were used (table 3). The BLAST runs were executed over a grid system installed at the Department for Genome Orientated Bioinformatics of the Technical University of Munich. The system was configured so that the execution takes place on only one workstation with no other jobs running.

BLAST version was blastall 2.2.14. PROMPT version was 0.7. The workstations are Pentium 4 with 2.66 GHz and 2 gigabytes of memory under Linux with Fedora core 5 as operating system.

3.1.3.3 PROMPT Fast Mapping

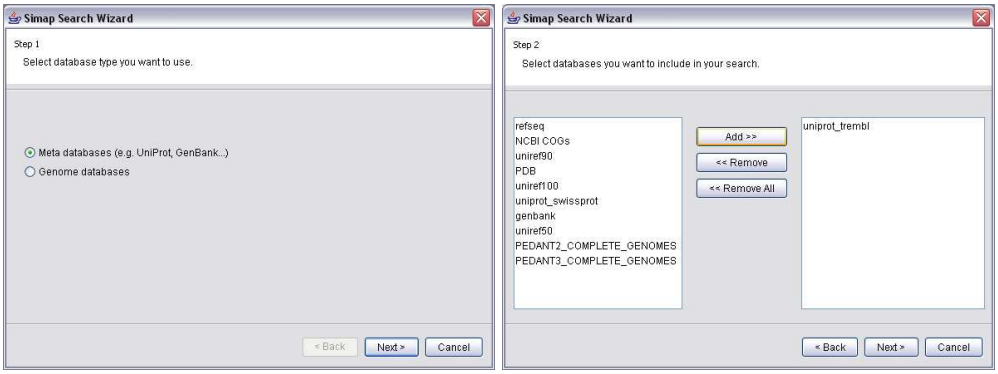
Unlike BLAST, the runtime of which depends on the size of the query sequence set and the size of the database you search against (figure 4), the runtime of PROMPT Fast Mapping only depends on the size of the query sequence set and slightly on the heuristic method you choose. As described above, PROMPT Fast Mapping utilises the SIMAP database, which contains pre calculated sequence homologies. If you query one of those sequences, it does not matter if you search against PDB (currently 27 369 sequences/21 158 580 amino acids in SIMAP) or UniProt-TrEMBL (currently 3 003 420 sequences/1 068 499 857 amino acids). It will take the same time because it is just another database query. This makes it possible to perform mapping tasks.

3.2 Implementation

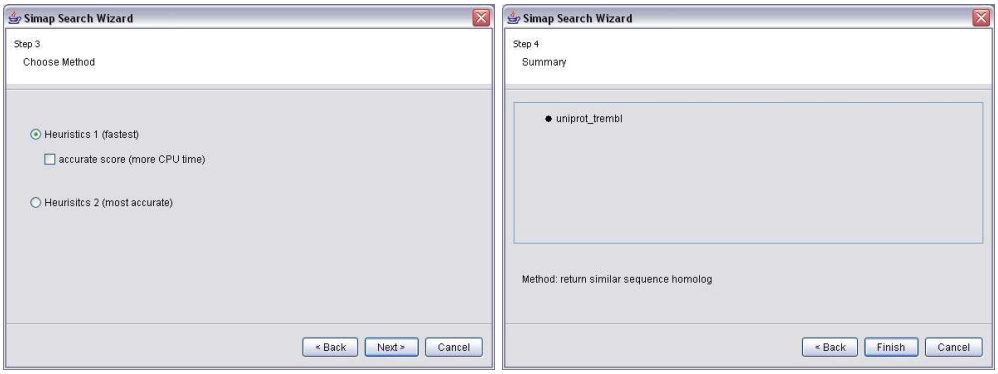
The PROMPT Fast Mapping method is implemented as part of the PROMPT framework. Like the framework itself, the Fast Mapping is written in JAVA. The main class, which interacts with the framework is `FastMapper`. This class takes one set of inputs, the set of query sequences. So it implements the `IEngine1`interface. It also implicitly implements the `IJobTemplate` interface which contains the `run()` method. Whenever an engine (input, visualisation or test) is called, the `run` method is invoked. For a detailed description of the framework refer to appendix A. The Fast Mapping consists of mainly three classes:

FastMapper

This is the main class. It is the engine class that implements the `IEngine` interface and is embedded in the processing layer of PROMPT. All user interactions are handled in this class. Upon calling this class the `run()` method is invoked. First, the class retrieves all available databases from SIMAP using the web services. Then a wizard like dialog will be opened which enables the user to select settings regarding the mapping, such as databases to search against and heuristic to apply to sequences that are not found. Figure 3 shows the four screens of the dialog. In figure 3 b. the screen for the meta database selection is shown. A smilar screen would pop up for the genome database selection. When the user closes the dialog, the class will pass the settings and the query sequences to a helper class that creates a `MapSimapIterator` instance. By calling `getResultIterator()` in the helper class, the class FastMapper now has an iterator containing the results. Actually, the results are not yet included in the iterator, but are retrieved on each call of the iterator's `next()` method. Each call of `next()` will return a `HashMap` object which holds results for a respective query sequence. The results are written to a tab delimited string. After all sequences are processed (no more iteration in the iterator), the mapping is finished and the results are available by the `getResultsAll()` method that returns an array of `IEngineResult` objects. In addition to the actual mapping results, a second result is created that contains statistical information about the mapping like search time, databases used and number of query sequences.



(a) step1 (b) step2



(c) step3 (d) step4

Figure 3 By using the PROMPT GUI, the user can select settings using this dialog. The first step (figure a) prompts the user to select the type of databases the query should be mapped against. After choosing, the dialog will show the selection for meta databases (like UniProt, PDB, UniRef...) or genome databases. In figure b, the selection for meta databases is shown. The next step is the selection of the heuristic to apply to sequences that are not found in SIMAP (figure c). A summary shows the settings made in this dialog (figure d)

MapSimapIterator

This class is an iterator that contains the mapping results. After the FastMapper class has passed all the query sequences and search settings, the iterator retrieves the result hit by calling the `getHomologs()` method in `Caller` on each call of the `MapSimapIterator` `next()` method.

Caller

This class handles all connections to the SIMAP database using web services. The method `getHomologs()` will return the result hit for a specific sequence that is passed to this method as parameter. Beside that, an instance of the web service and the heuristic that should be used are passed to this method. This method will query the sequence in SIMAP. If the sequence cannot be found, the most similar sequence is looked up and the homologs of this sequence is queried. If this happens, the chosen heuristic is applied to the result and the result of the heuristic is returned by the `getHomologs()`method. The detailed workflow of this class is shown in detail in figure 2.

4 Results

The important questions concerning the new method described here are: ”How fast and how accurate is it”. The PROMPT Fast Mapping method is compared to a BLAST search with respect to runtimes and accuracy. As described in Methods, BLAST was used as a reference to assess the quality and runtime of PROMPT Fast Mapping because of its extensive use by molecular biologists. BLAST was evaluated with test sets against various database sizes (table 3). In figure 4 the runtimes of BLAST runs of the seven subject databases are shown. The accuracy is also evaluated comparing results from PROMPT Fast Mapping to the result of BLAST.

4.1 Runtimes

4.1.1 BLAST

In figure 4 the runtimes of BLAST searches against databases of different sizes are shown. A strong linear correlation (Pearson’s correlation coefficient: 0.9996) between database size with respect to amino acid content and runtime is indicated.

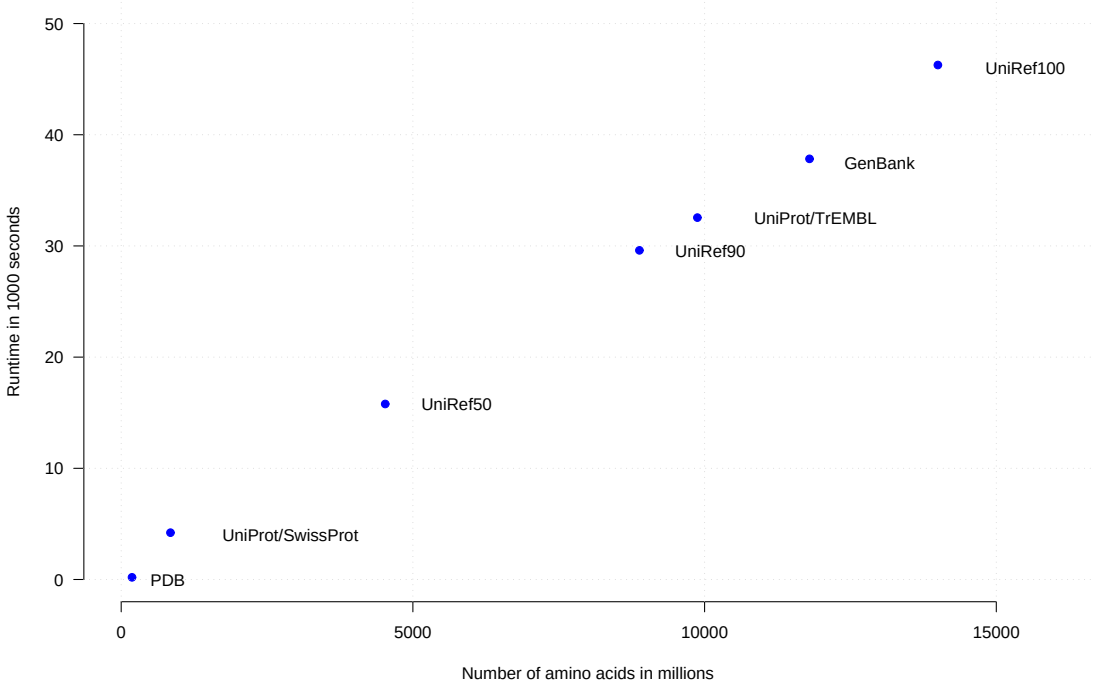


Figure 4 Runtimes of BLAST runs. Blasting 1000 sequences against different databases with different sizes. See table 2 for exact sequence and amino acid count. A Pearson correlation coefficient is 0.9996 indicates almost perfect linear correlation.

4.1.2 PROMPT Fast Mapping

The runtimes for the different databases mentioned in table 3 are shown in figure 4. For every database a mapping using PROMPT Fast Mapping is conducted using both heuristics. The runtimes of the PROMPT Fast Mapping runs for all the databases, using the query dataset of randomly extracted sequences with no mutation, in table 3 are shown in figure 5. Also included in this figure are the BLAST runtimes already shown in figure 4. As already stated, the runtime of a Fast Mapping run only depends on the size of the query sequence set and on the availability of sequences in SIMAP. The red or green points in figure 5 do not exactly lie on a

straight line. This difference in runtime derives from factors like network status or traffic. The exact runtimes are shown in table 5.

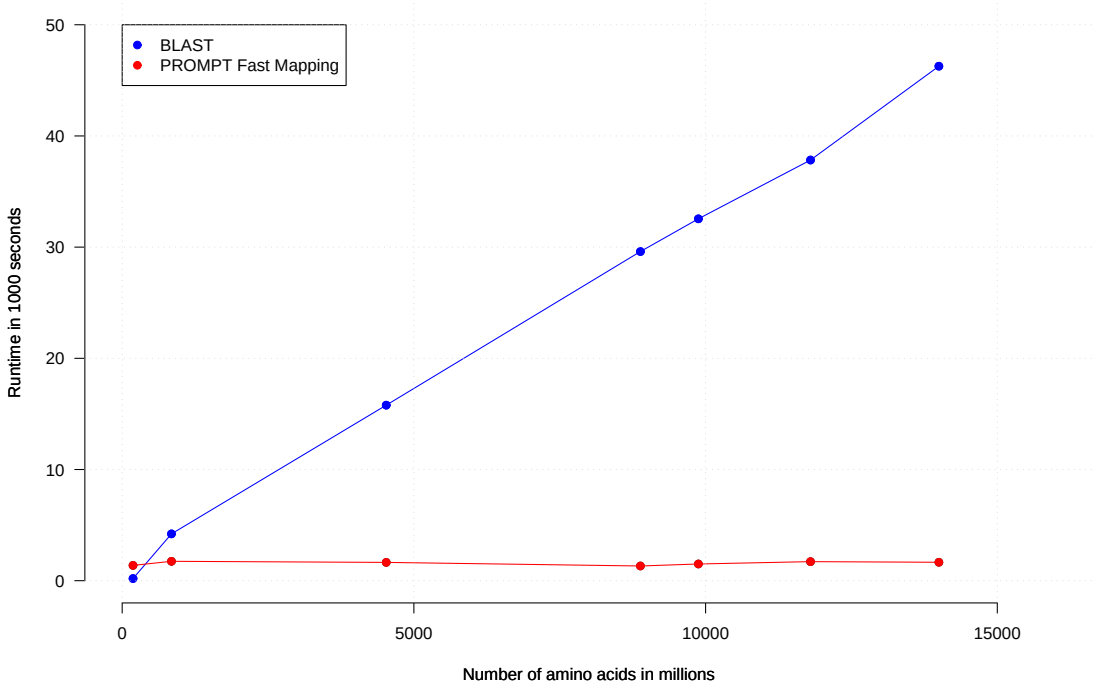


Figure 5 Runtimes of BLAST runs compared to those of PROMPT Fast Mapping runs. In this case the red line shows the runtimes for a query dataset of sequences that were extracted from the TrEMBL dataset by random

The runtime show that the speed gain is higher the larger the database is, the mapping is performed against. For the largest database (UniRef100) the BLAST run takes 12 hours and 51 minutes (mean) and the PROMPT run (with the faster heuristic) only 27 minutes. That is 28 times faster.

For the mutated dataset the runtimes are shown in figure 6. Again, the runtime of the mapping against the different sizes of databases is (nearly) constant. Although it is higher because of the time demanding seed lookup. The mean time of 2 hours and 52 minutes for the faster heuristic is still a 4,5 times faster than the BLAST runtime

Database	Runtimes			
	BLAST	**non mutated**	**mutated, heuristic A**	**mutated, heuristic B**
PDB	00:03:14	00:22:56	03:09:18	02:53:32
SwissProt	01:10:11	00:29:03	03:11:38	02:58:46
TrEMBL	09:02:25	00:27:25	03:14:08	02:57:23
UniRef50	04:23:06	00:22:02	03:07:11	02:44:22
Uniref90	08:13:23	00:25:04	03:09:45	02:52:02
UniRef100	12:51:12	00:28:34	03:12:34	02:57:48
GenBank	10:30:33	00:27:36	03:10:22	02:45:32

Table 5 In this table all runtimes are shown. The runtimes of the mutated dataset using heuristic A is slightly higher than those of heuristic B because of the FASTA search that is performed after the retrieval of the best 50 hits to the seed sequence.

of the largest dataset of the evaluation databases used. The green line shows the runtimes for the more accurate heuristic. These are higher because of the FASTA search that is performed.

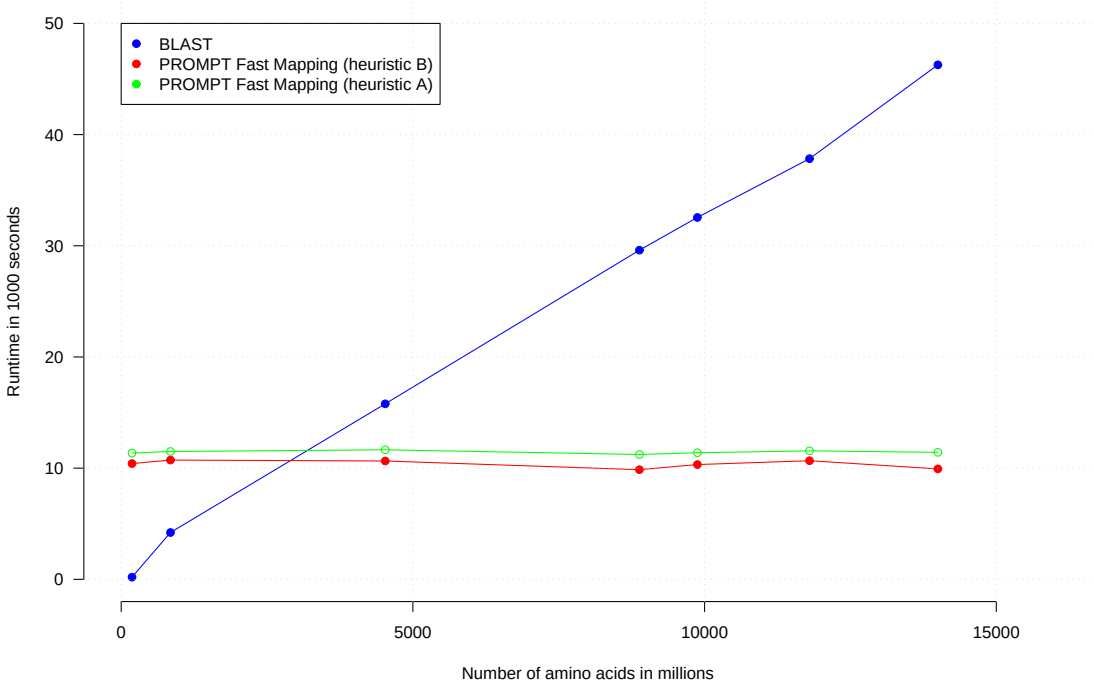


Figure 6 Runtimes of BLAST runs compared to those of PROMPT Fast Mapping runs. In this case the red and green lines show the runtimes for a query dataset of sequences that were extracted from the TrEMBL dataset by random and mutated as described in 3.1.3

4.2 Seed lookup

For sequences that are not contained in SIMAP, the most similar sequence is looked up and for this a homologous will be searched in SIMAP.

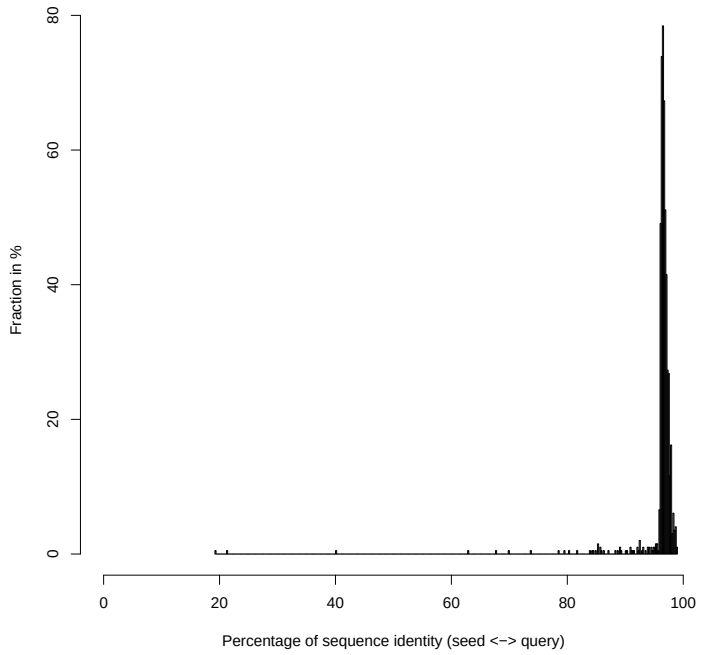


Figure 7 Result of the comparison of the seed sequence with the query sequence. The diagram shows the sequence similarity of seed and query sequence. The identity is based on the Smith-Waterman algorithm and calculated using JAligner. 971 of 1000 (97.1%) sequence pairs have a similarity of more that 90% with a small standard deviation of 0.74

.

This is the basis of the PROMPT Fast Mapping method. The assumption is that a similar sequence produces a similar hit as the original query sequence. To assess the quality of the seed sequences and the accuracy of the seed lookup procedure, the query sequences are compared with the respective seed sequence. For the sequences in the mutated dataset, which consists of sequences chosen by random from the

UniProt/TrEMBL database and then mutated, the query sequences are retrieved. For each pair of sequences, an alignment is calculated using JAligner[1]. JAligner is an implementation of the Smith-Waterman algorithm for local sequence alignments. Figure 7 shows the distribution of sequence identities. The mean of identities is 96.78%. 971 sequences (97.1%) have a similarity greater than 90%. But there are also some pairs with a small identity. A reason for these small identities is demonstrated in figure 8. An alignment shows the difference of a query/seed pair with low identity. The seed sequence contains the original query only as a substring. In this section, the quality of the seed sequences is analysed. The next section will show the result of the overall accuracy of PROMPT Fast Mapping.

[1] http://jaligner.sourceforge.net

```
QUERY  --------------------------------------------------

SEED   MSNEVEQKKNIKTINDLPGISQTVINKLIEAGYSSLETLAVASPQDLSVAAG

QUERY  --------------------------------------------------

SEED   IPLSTAQKIIKEARDALDIRFKTALEVKKERMNVKKISTGSQALDGLLAGGI

QUERY  ----------------TQLCHQLSVNVQLPPEKGTLSGKAVYIDTEGTFRWE
                       |||||||||||||||| ||||||||||||||||
SEED   ETRTMTEFFGEFGSGKTQLCHQLSVNVQLPPEKGGLSGKAVYIDTEGTFRWE

QUERY  RIENMAKALGLDIDNVMNNIYYIRAINGDHQIAIVDDLQELVSKDPSIKLIV
       |||||||||||||||||||||||| ||||||||||||||||||||||||
SEED   RIENMAKALGLDIDNVMNNIYYIRAINTDHQIAIVDDLQELVSKDPSIKLIV

QUERY  VDSVTSHFIAEYPGRENLAVRQQKLNKHLHQLTRLAEVYDIAVRI-------
       |||||||| |||||||||||||||||||||||||||| |
SEED   VDSVTSHFRAEYPGRENLAVRQQKLNKHLHQLTRLAEVYDIAVIITNQVMAR

QUERY  --------------------------------------------------

SEED   PDMFYGDPTVAVGGHTLYHVPGIRIQLKKSRGNRRIARVVDAPHLPEGEVVF

QUERY  ------------

SEED   ALTEEGIRDAEE
```

Figure 8 An alignment of seed and query sequence with an identity of 40.1%. The length differs because only a substring of the query is contained in the seed sequence

4.3 Accuracy

To assess the quality of PROMPT Fast Mapping the results are compared to the results of a BLAST run. The sequences from the non mutated dataset were mapped against the UniRef database (UniRef100) using PROMPT and BLAST. For the PROMPT search, the 10 best homologs were retrieved from SIMAP to check on which position in the PROMPT Fast Mapping result the best hit from BLAST is shown. The search was performed for both heuristics. For heuristic B, the faster heuristic, of 1000 sequences in the dataset, 994 sequences produce exactly the same hit in PROMPT Fast Mapping and in a BLAST search. The remaining 6 sequences are shown on position 2 on the Fast Mapping result. So 99.4% of the results produced by PROMPT Fast Mapping are equal to the results of a BLAST search. For the more accurate heuristic, all query sequences (100%) produce the same hit in BLAST and PROMPT so this heuristic is as accurate as the BLAST search. So even the ”bad” seed sequences mentioned in the previous section, produce hits, that are equivalent to those of a BLAST search.

4.4 Availability

PROMPT Fast Mapping will be available in different formats to make it suitable for a broad range of use cases. Mainly it is part of the PROMPT framework. PROMPT is available for freely available for academic users at http://webclu.bio.wzw.tum.de/prompt/. Included is the PROMPT GUI where Fast Mapping can be executed in a user friendly environment but also the possibility to run beanshellscripts without using the GUI. A web interface to the Fast Mapping method is also available at http://webclu.bio.wzw.tum.de/prompt/.

5 Discussion

The aim of this work was to develop a tool for fast similarity searches. Using a database of pre calculated sequence homologies and heuristic methods, the time complexity of the task of finding sequence homologies to a given sequence is reduced. Although, because of the longer runtime compared to a BLAST run, this method is not suitable for small databases like PDB, it speeds up the process of similarity search by a factor up to 27 or more. The runtimes in figure 5 are those of a BLAST run and those of the PROMPT Fast Mapping method. The red line (for the runtimes of PROMPT Fast Mapping) crosses the blue line (for the runtimes of a BLAST run) at a database size of approximately 350 million amino acids. This mean that using PROMPT Fast Mapping makes sense if the database you search against is large than 350 million amino acids presuming that all sequences in the query sequence set have a corresponding sequence in the database of pre calculated sequences. The only database in the evaluation environment described in section 3.1.3 where the PROMPT Fast Mapping does not make sense to use is PDB. All searches against databases that are lager than 350 million amino acids show an advantage by using the PROMPT Fast Mapping method. Sill presuming that no heuristic is applied. If a sequence is not included in the SIMAP database, the runtimes of the PROMPT Fast Mapping runs are higher. In figure 6, the runtimes for a query set of mutated sequences are shown. Querying these sequences takes longer because of the seed lookup. The more a sequence differs from any sequence in SIMAP, the longer the lookup for a similar sequence takes. In this case, the search is still sped up by a factor of 4.7 (PROMPT: 2:45:32; BLAST: 12:51:12). For the not mutated query sequence set, the maximum factor is 27 (PROMPT: 00:28:34; BLAST: 12:51:12). Regarding the growth of the public available databases, the speed up factor will grow with the databases. No matter how fast a database will

grow, the runtime of PROMPT Fast Mapping is always the same. The evaluation sets used show two possible extremes. One sequence set contains only sequences that are included in SIMAP and therefore need no seed lookup. The other set contains only sequences that could not be found in SIMAP and need a seed lookup. For the first set, the average time for one sequence is 1.6 seconds, for the other set, the average time is 10.3 seconds. If the degree of mutation is similar to that was used for evaluation (see section 3.1.3 for a detailed description of the mutation), the runtime for 1000 sequences is between 02:45:32 (the mean runtime of the mutated sequence set) and 00:28:34 (the mean runtime of the not mutated sequence set). The accuracy of PROMPT Fast Mapping is rather good compared to a reference BLAST run. 99.4% of the sequences produce the same hit in PROMPT and BLAST. The seed sequence lookup procedure works well for the type of mutation that is used for this evaluation. Nevertheless it has to be said that the more different a sequence is to the sequences in SIMAP, the longer the search will take. With the current implementation of the SIMAP web service, a timeout will occur when sequences are search that are very unsimilar to sequences in SIMAP. This problem has to be addressed by the curators of SIMAP. New hardware with enough memory to store all the indexed fragments of the similarity search will eliminate the timeout problem of some search but would also speed up the seed lookup procedure significantly and therefore would speed up the whole mapping process.

The integration of the PEDANT database into PROMPT enables a user to access a database of over 500 different genomes and the corresponding rich annotations. In combination with PROMPT Fast Mapping, a user can instantly get all information annotated to a specific sequence by mapping the query sequence to the PEDANT database using PROMPT Fast Mapping and then use the result as a new input in PROMPT. So the user can now apply all analyses or comparisons to the previously

mapped sequences. This makes PROMPT an outstanding tool for comparative proteomics.

6 Conclusion

The implementation of PROMPT Fast Mapping is a powerful, fast and accurate tool for mapping tasks. To perform those time consumptive tasks in reasonable time, strong computing facilities are necessary. With the new method mapping times are no longer dependent on the user's computing power. Slow computers with not much memory are capable of mapping large datasets. Additionally, client side pre processing like formatdb for BLAST is not necessary any longer. As part of the PROMPT framework the user can take advantage of PROMPT's comfortable graphical user interface. Also available is a web front end at http://webclu.bio.wzw.tum.de/prompt/. The integration of the PEDANT database in combination with PROMPT Fast Mapping offers a rich platform for bioinformatic analyses.

7 Appendix

A The PROMPT framework

In research labs throughout the world, proprietary applications for mapping or comparison tasks are written in form of one time use and small scope manner. Tools are written for a specific task allowing few changes in its application and are therefore rewritten to fit the need for a certain research task. For example, comparing the hydrophobicity of sequences or retrieving sequences for specific protein identifiers. To address this problem, Schmidt and Frishman (2006) developed a bioinformatics framework for large scale protein comparison, PROMPT.

Figure 9 Screenshot of the PROMPT GUI

The PROMPT framework is a comprehensive software environment to address a wide spectrum of routine tasks in comparative proteomics. It enables the user to compare arbitrary amino acid sequences, correlate sequence features and perform mapping tasks in a user friendly environment. Its flexible implementation makes it suitable for wide range of comparative problems but also offers space for user defined extensions like input sources and proprietary analysis methods.
The PROMPT data processing workflow is based on three layers:

1. Input layer
2. Processing layer
3. Visualisation layer

These layers are connected via well defined interfaces which allow, in combination with methods for cross layer communication, to use an algorithm with different input sources and a visualisations on different algorithm results.
In any layer, data can be exported. Sequences and annotations from the input layer, results of respective analyses from the processing layer and graphs and plots can be exported from the visualisation layer.

Input layer The input layer handles all types of input. At the moment imports are available for various flat files like FASTA, GenBank, EMBL and UniProt. Additionally, data can be imported from MIPS[2] databases like PEDANT or CYGD[3]. All implementations of the `ISeqProperties` interface are able to provide sequences and therefore usable for any method that need sequences like mapping tasks or sequence analyzation. Therefore it defines methods like `getSequences()` or

[2]http://mips.gsf.de

[3]Comprehensive Yeast Genome Database, http://mips.gsf.de/genre/proj/yeast/

`getSequencesNames()`. These methods are called by classes of the processing layer. Figure 10 shows selected inputs and their connections.

There are also methods for determining properties that can be calculated directly from the protein sequence like mass, isoelectric point or sequence length. These properties are calculated mainly using BioJava[4]. Sequences and annotations which are available in major public databases can be retrieved by their respective protein identifier. If a list of UniProt or GenBank identifiers is provided, the corresponding information is downloaded automatically using SeqHound (Michalickova *et al.*, 2004) while importing the identifier list. Besides the flat file and database imports, the user can import data in PROMPT's custom XML format. Arbitrary data, numerical as well as nominal data can be stored in this XML scheme. Figure 11 shows a sample file. Below the root element (`dataset`) any number of `property` elements can be inserted providing any type of information tied to a sequence. Numerical attributes, like the number of predicted transmembrane segments, molecular weight or hydrophobicity, and nominal attributes like functional categories or EC numbers. The type is indicated by the property attribute `type`, the name of the property element is indicated by the attribute name. A special property of type (`setdef`) provides the respective amino acid sequences of the protein identifiers included and thus provides the mapping from the identifiers to the sequences.

[4]http://www.biojava.org

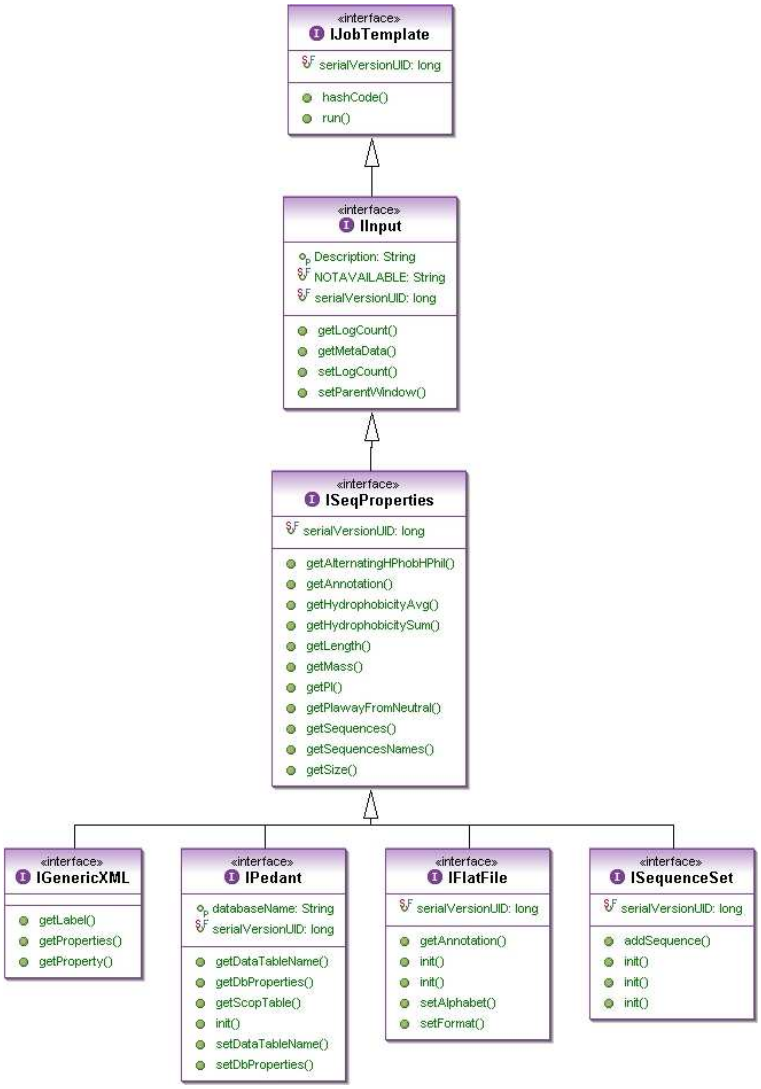


Figure 10 Input interfaces of PROMPT. In this diagram only a selecting of the available interfaces is shown. Like classes in PROMPT that need to process data, the top interface is the `IJobTemplate` interface. The `IInput` interface is the base interface for all input methods. For all inputs that should provide sequences, the `ISeqProperties` interface contains the necessary methods. In the hierarchy below this interface are the interfaces for the different types of input. PROMPT Fast Mapping works with all inputs shown here.

```
<dataset label="Escherichia_coli_k12">
  <property id="setdef" type="setdef">
    <input id="P68191" value="MKSNRQARHIL...">
    <input id="P00882" value="MTDLKARRLRI...">
    ...
  </property>
    <property id="transmembrane segments" type="numeric">
    <input id="P68191" value="0">
    <input id="P00882" value="6">
    ...
  </property>
    <property id="funcat" type="symbolic">
    <input id="P68191" value="04.02">
    <input id="P00882" value="01.01.01.02">
    ...
  </property>
</dataset>
```

Figure 11 Custom XML format of PROMPT.

Processing layer Analyses can be performed on any type of annotated information contained in input files, but also on characteristics that can be calculated from the amino acid sequence like mass, isoelectric point or hydrophobicity. The characteristics are calculated mainly using BioJava. For annotated information PROMPT contains a set of generic analysis methods to compare or correlate nominal and numerical attributes. These are basically

- nominal feature comparison between two sets
- nominal feature enrichment between a set and a subset
- numeric feature comparison
- numeric feature correlation

To determine if there is a significant difference in input sets with respect to features of interest, statistical tests are applied automatically by PROMPT but can also be applied manually by the user. For this, PROMPT contains a wide range of different established tests that can easily be applied via the PROMPT GUI. All tests are performed by the open source project "R"[5] or its commercial counterpart SPLUS, depending on availability.

Visualisation layer The best presentation of an analysis result is a graphical representation. The results can be examined in different plots and figures, depending on the analysis. For visualisation, PROMPT uses the facilities provided by R but also interactive figures can be used created by JFreeChart[6]. All figures can be exported in different formats like bitmap orientated portable network graphics (png) or scalable formats like encapsulated postscript or windows meta format. Figures can also be printed out of PROMPT directly.

Overview The interface layout of the PROMPT framework allows user-defined extensions for proprietary methods. These extensions are developed in the form of Java classes that extend the respective interface to create new types of input, processing and visualisation methods. In figure 12, a class diagram shows an overview of the main interfaces.

[5] http://www.r-project.org
[6] http://www.jfree.org

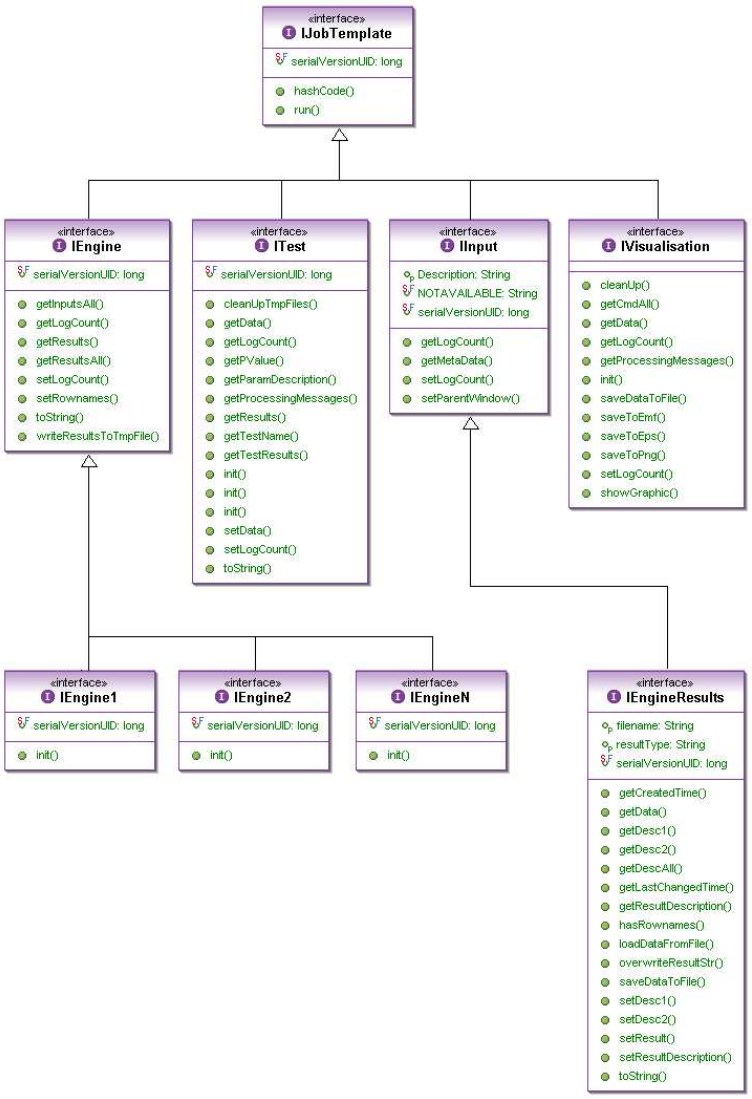


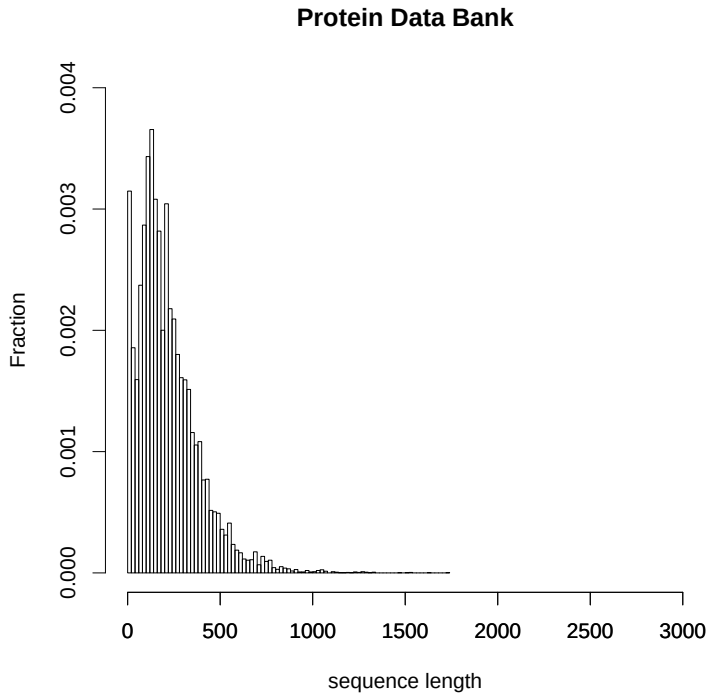
Figure 12 Overview of the PROMPT main interface structure

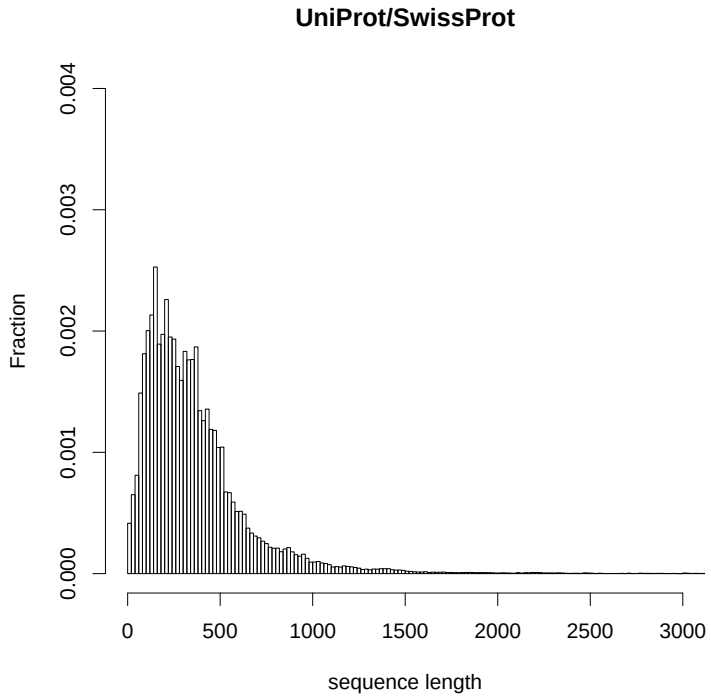
The interfaces `IInput`,`ITest`,`IEngine` and `IVisualisation` all extend the `IJobTemplate` interface. This interface contains the method `run()` that executes the respective class. So in the `run()` method, the actual work is done. This method also provides an interface to grid systems or distributed computing.

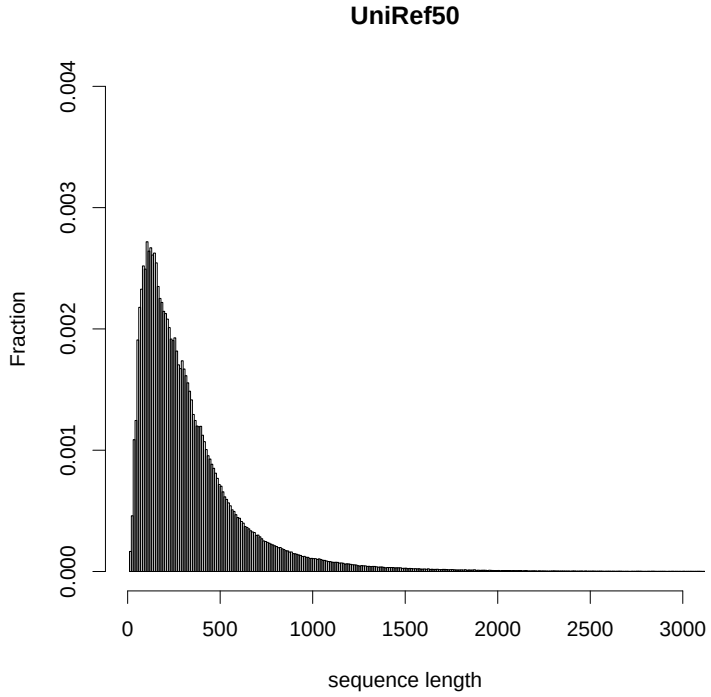
B The SIMAP database

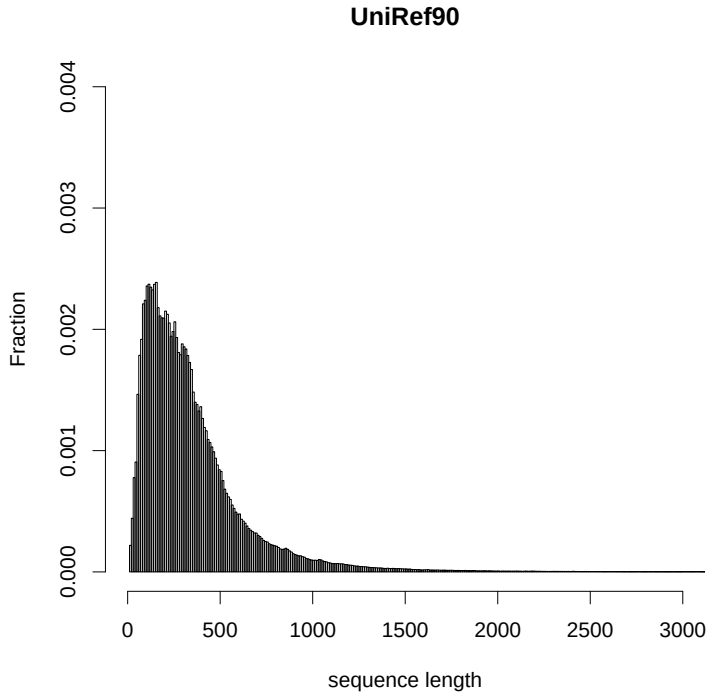
Sequence comparisons are a fundamental task in comparative genomics and proteomics. Comparing (aligning) two sequences can be done in matter of seconds or even milliseconds but at large scale for example aligning a sequence to a whole genome or a whole database like UniProt/TrEMBL, containing around 4 million sequences, this task can be very time consuming. A solution can be a database which contains pre calculated sequence alignments. ”Aligning” a sequence with sequences from this database results in just a database query. Rattei *et al.* (2006) implemented such a database, called SIMAP (the similarity matrix of proteins). At the moment this database contains about 6 million sequences which are aligned with each other. If a new sequence is added, sequence alignments with all other sequences are calculated by using FASTA and the result is stored in a database. These calculations are performed using a cluster and a grid system installed at the chair for Genome Orientated Bioinformatics of the Technical University of Munich but also a distributed computing project called BOINC (Berkeley Open Infrastructure for Network Computing).

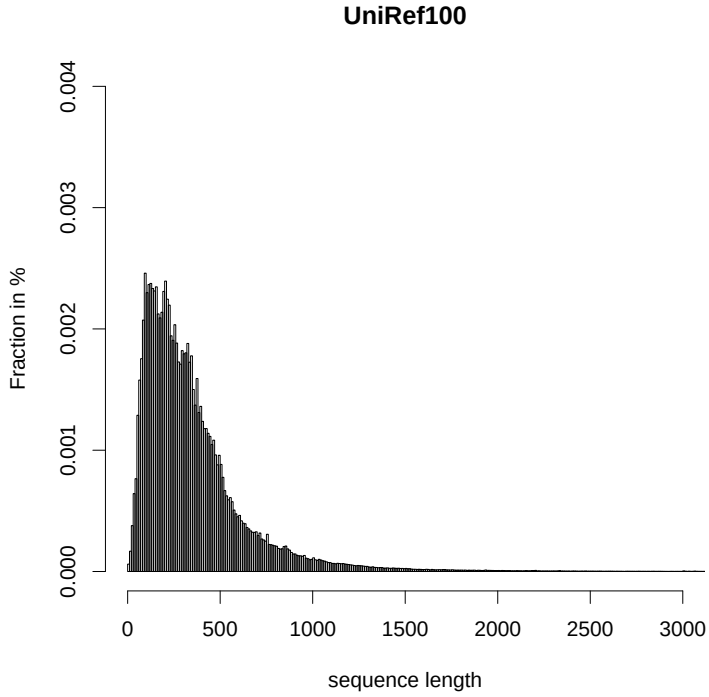
C Length distributions

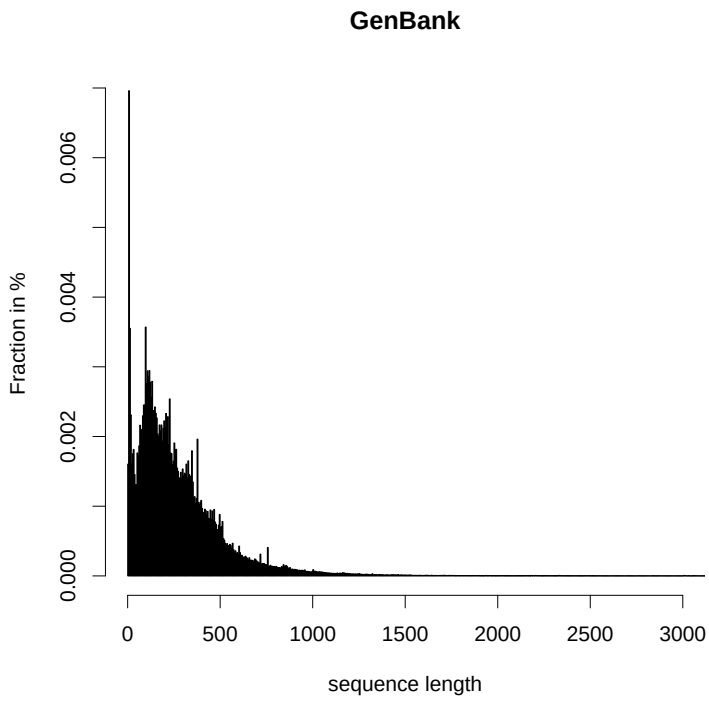





UniRef50
Fraction
0.000
0.001
0.002
0.003
0.004
0
500
1000
1500
2000
2500
3000
sequence length


UniRef90
Fraction
0.000
0.001
0.002
0.003
0.004
0
500
1000
1500
2000
2500
3000
sequence length


UniRef100
Fraction in %
0.000
0.001
0.002
0.003
0.004
0
500
1000
1500
2000
2500
3000
sequence length


GenBank
Fraction in %
0.000
0.002
0.004
0.006
0
500
1000
1500
2000
2500
3000
sequence length

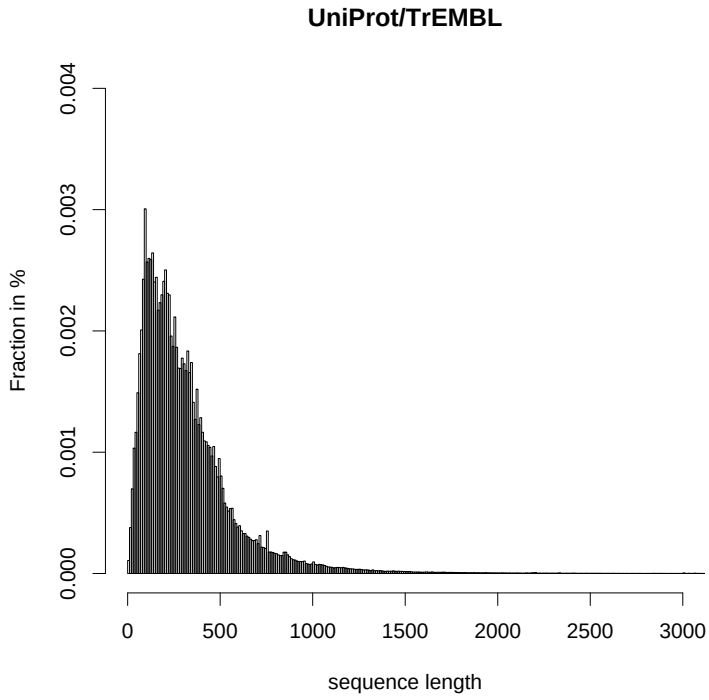


Figure 13 Length distribution of database files downloaded from the respective download sites.

D Database download links

- GenBank
  ftp://ftp.ncbi.nih.gov/genbank/
- UniProt (SwissProt, TrEMBL)
  ftp://ftp.expasy.org/databases/uniprot/current_release/knowledgebase/complete/
- UniRef (UniRef100, Uniref90, UniRef50)
  ftp://ftp.expasy.org/databases/uniprot/uniref/

References

Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.

Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. 5th IEEE/ACM International Workshop on Grid Computing.

Benson, D., Karsch-Mizrachi, I., Lipman, D., Ostell, J., *et al.* (2005). Genbank. *Nucleic Acids Research*, **33**(Database Issue), 34–38.

Berman, H., Westbrook, J., Feng, Z., Gilliand, G., *et al.* (2000). The protein data bank. *Nucleic Acids Research*, **28**(1), 235–242.

Collins, F., Morgan, M., and Patrinos, A. (2003). The human genome project: Lessons from large-scale biology. *Science*, **300**(5617), 286–290.

Draghici, S., Sellamuthu, S., and Khatri, P. (2006). Babel's tower revisited: a universal resource for cross-referencing across annotation databases. *BMC Bioinformatics*, **22**(23), 2934–2939.

Elahi, E. and Ronaghi, M. (2004). Pyrosequencing: a tool for dna sequencing analysis. *Methods in molecular biology*, **255**, 211–219.

Frishman, D. and Mewes, H. (1997). Protein structural classes in five complete genomes. *Nature Structural Biology*, **4**(8), 626–628.

Frishman, D., Mokrejs, M., Kosykh, D.and Kastenmuller, G., Kolesov, G., Zubrzycki, I., *et al.* (2003). The pedant genome database. *Nucleic Acids Research*, **31**(1), 207–211.

Kent, J. (2002). Blat - the blast-like alignment tool. *Genome research*, **12**(4), 656–664.

Michalickova, K., Bader, G. D., *et al.* (2004). Seqhound: biological sequence and structure database as a platform for bioinformatics research. *BMC Bioinformatics*, **3**, 32.

Pearson, W. (1990). Flexible sequence similarity searching with the fasta3 program package. *Methods in molecular biology*, **215**(3), 403–410.

Rattei, T., Arnold, R., Tischler, P., Lindner, D., Stumpflen, V., and Mewes, H.-W. (2006). Simap: the similarity matrix of proteins. *Nucleic Acids Research*, **34**(Database Issue), 252–256.

Riley, M., Schmidt, T., Artamonova, I., Wagner, C., Volz, A., *et al.* (2006). Pedant genome database: 10 years online. *Nucleic Acids Research*, **35**(Database issue), 354–367.

Rivest, R. (1992). The md5 message-digest algorithm. Network Working Group, Request for Comments: 1321.

Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., *et al.* (2004). The funcat, a funtional annotation scheme for systematic classification of protein from whole genomes. *Nucleic Acids Research*, **32**(18), 5539–5545.

Schmidt, T. and Frishman, D. (2006). Prompt: A protein mapping and analysis tool. *BMC Bioinformatics*, **7**(1), 331–346.

Smith, T. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**(1), 195–197.

Stephens, S., Chen, J., and Thomas, S. (2003). Odm blast: Sequence homology search in the rdbms. http://www.oracle.com/technology/industries/life_science/pdf/ls_ieee_blast.pdf.

Tatusov, R., Fedorova, N., Jackson, J., Jacobs, A., *et al.* (2003). The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, **4**, 41.

Weerawarana, S., Curbera, F., and Leymann, F. (2005). *Web Services Platform Architecture*. Prentice Hall.

Wilson, C. A., Kreychman, J., and Gerstein, M. (2000). Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *Journal of Molecular Biology*, **297**(1), 233–249.

Wu, C., Apweiler, R., Bairoch, A., Natale, D., *et al.* (2006). Universal protein resource (uniprot): an expanding universe of protein information. *Nucleic Acids Research*, **34**(Database Issue), 187–191.