Diploma Thesis

A module for semi-automated annotation of megabase-sized DNA sequences by homolgy search

Stefan Michael Schuster

January 6, 2008

Supervisor:
Prof. Dr. Bernhard Haubold
FH Weihenstephan
Fakultät Biotechnologie und Bioinformatik
85350 Freising
Germany

Supervisor:
Dr. Jannick D. Bendtsen
CLC bio A/S
Gustav Wieds Vej 10
8000 Aarhus C
Denmark

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 .....

.....
Stefan Michael Schuster

1 Summary

Costs for sequencing DNA sequences are decreasing steadily. However, manual annotation of this sequences is still comparatively costly. Therefore, there is a increasing demand for software that allows a largely automated annotation process. In this thesis a software module was developed that extends an existing bioinformatics software with the functionality of semi-automated annotation.
The used approach for the annotation of megabase-sized DNA sequences is mainly based on a search for homologous protein sequences in the RefSeq database. For the homology search itself the freely accessible BLAST server at the NCBI is used. However, the search for the megabase-sized sequence is not done in one huge BLAST request. The sequence is rather split in overlapping subsequences, which are submitted to the BLAST server, and a BLASTx search is performed for each subsequence separately. The first reason why this is done is a restriction by the server: Requests for sequences considerable longer than 50kbp can not be processed but are aborted with an error message. The second reason is a notable increase of speed as several requests can run parallel. The returned results of all searches are merged and displayed graphically. All sequences in the database that had a significant hit in one of the BLAST searches are downloaded. Depending on the alignments produced by BLAST and the settings for filtering, the annotations of these sequences are transfered to the unknown sequence. The newly annotated sequence is shown in a graphical view that provides several tools for postprocessing.

Contents

2 Introduction

2.1 Document Conventions

In this thesis, certain words or phrases are written in italic type, like *ab initio gene finding*. This indicates that a description can found in the appendix, either in the glossary or in the list of acronyms and abbreviations. A word written in small capitals, like BLASTOUTPUT, represents an object in Java.

2.2 Sequence Annotation

The fundamental information of a sequence is obviously provided by its primary structure. For *Deoxyribonucleic Acid (DNA)* sequence this is simply a succession of the the four bases adenine, cytosine, guanine, and thymine. With a sequencer, this data can be extracted from a *DNA*-molecule and stored in files or databases. But without additional information, it is almost impossible for a human to gain any knowledge out of a file containing merely As, Ts, Cs, and Gs. Therefore there is high demand for meta informations and annotations are added to the sequences. An annotation consists of two elements, a textual description and the region on the sequence for which this description stands. For *DNA* sequences, an annotation can specify amongst others genes, coding sequences, regions that code for proteins, *Open Reading Frame (ORF)s* or *Single Nucleotide Polymorphism (SNP)s* . For proteins this can for instance be structural information like a helical region or a beta strand or information about the molecular function. In principal, all relevant information about a sequence can be added as an annotation. But in order to standardize the annotations, there are projects like Gene Ontology [1] that provide a controlled vocabulary.

2.2.1 Sequencing

Sequencing means to read the primary structure of a biological molecule. The costs are decreasing steadily, as there are new methods invented and the existing methods are permanently optimized.

DNA-Sequencing

In 1977 W. Gilbert and A. M. Maxam published a method for *DNA*-sequencing that is based on chemical modification of *DNA* and subsequent cleavage at specific bases [2]. F. Sanger published his method for sequencing also in 1977 [3] which is based on chain-termination caused by labeled dideoxynucleotides. For their work on sequencing, Sanger and Gilbert won both a fourth of the nobel-prise in chemistry in 1980. Since then, great improvements in reducing the time and costs were made by automating the process to a very high degree. The first automated sequencers in the 1980's had a throughput of several kilobases a day, nowadays however, actual sequencers [1] can process up to several megabases in the same time.

2.2.2 Gene Finding

Genes are the part of the *DNA* sequence that code for proteins and are therefore very interesting regions for sequence analysis. Hence, a feasible first step for an annotation process is to find this regions. There are two principal approaches for the search of genes, namely *comparative*

[1] like the 3730xl from Applied Biosystems, according to http://docs.appliedbiosystems.com/pebiodocs/00113233.pdf

gene finding and *ab initio gene finding* [4]. Several programs for computational gene finding are available, all are using one or both of this approaches. A comparison of the current methods was made in [5].

ab initio gene finding

These methods, also referred to as intrinsic approach, are based on the primary structure itself. For procaryotic sequences, a god way to start is to search for *ORFs*, as in these sequences a gene is only one coding region without any interrupts. However, it is more difficult for eucaryotic sequences [6]. Due to the intron exon structure of genes in eucaryotic sequences the *ORFs* for the exons may be too short to be distinguishable from *ORFs* occurred by chance only. Therefore other methods are used for this sequences. For the transcription, translation and splicing processes there are specific signals. These are generally short sequences between 2 and 10 bp long that give the genomic apparatus the instruction to initialize these processes. Below are some selected signals:

- TATA-box in the promoter region of eucaryotic genes
- polyadenylation signal as part of the transcription termination
- stop codon as end of translation
- donor site and acceptor site for the spliceosomes

Also the content of the sequences is an indication whether a part of a sequence is coding or not. There is a difference in the frequency of oligomers between coding and non-coding regions and this difference is the most significant for hexamers [7]. With detailed information about this frequencies, Markov Models are created for an effective discrimination.

comparative gene finding

Comparative methods, also called extrinsic approach, are mainly based on a homology search to already known sequences. Obviously, only sequences with a known homologue counterpart can be annotated, but as the number of well annotated sequences is increasing this disadvantage becomes less important. Three types of methods are distinguished. The main difference between them is the type of sequence against the genomic sequence is searched:

- protein sequences
- cDNA or expressed sequence types
- another genomic sequence

The first part of the approach used in this thesis is similar to a simplified version of *comparative gene finding* with protein sequences.

2.3 BLAST

The *Basic Local Alignment Search Tool (BLAST)* [8] is a software package for the search for statistically significant similarities between sequences. It is very well documented. Articles [9], books [10] as well as online resources [11] provide a efficient source for detailed information on history, methods and advanced usage of *BLAST*. Therefore not all of this is described here but only the parts that are relevant for this project.

2.3.1 Function

BLAST compares a sequence, (the *query sequence*), against a database (the *BLAST database*) by creating and evaluating local alignments. The *BLAST database* consists of one ore more sequences and is preformated to accelerate the search. Very good local alignments are called an *high scoring segment pair (HSP)*. Each sequence in the *BLAST database* for which *BLAST* produced at least one *HSP* to the *query sequence* is a *hit sequence*.
Unlike the Smith-Waterman algorithm [12], *BLAST* does not guarantee to find the optimal local alignment. A comparison of the sensitivity and specificity of both algorithms was made by Shpear et al. [13]. The discrepancy between both methods is a result of the fact, that *BLAST* uses much faster *heuristic methods* instead of searching the whole search space between the sequences. But in a biological context not only the best alignment may be relevant. The important criteria is rather the statistically significance of the alignment, and this is what *BLAST* returns.
The *BLAST* algorithm is composed of 3 steps. Please note that *BLAST* uses some variations of the following steps. But for the sake of clarity the description here is slightly simplified, a more detailed description can be found in [10]. Figure 1 shows an overview of the algorithm.

Seeding

For a significant alignment it is very likely that it contains a perfect match of at least W contiguous letters. W is called the *word-size*, and all chains of letters with a length of W are called a *word*. *BLAST* is now searching for all *word-hits*, i.e. all perfect matches of size W. Nevertheless, some significant alignments without a perfect match may exist especially for protein sequences. Therefore, a *word-hit* is defined less restrict as mentioned above. In order to explain this, the term *neighborhood* has to be introduced. The *neighborhood* of a *word* is a set of other *words*. The criteria for a *word* to be in this set is that the score is greater or equal to a threshold T when compared to the original word via the *scoring matrix*. Now, a *word-hit* can be either an exact match to a *word* or to a *word* in its *neighborhood*.

Extension

The extension step consists of a loop with two commands. First, the *word-hit* is extended in both directions. Second, the score for an alignment of the two segments is calculated. The algorithm terminates if the maximum score did not increase for a defined number of iterations. The alignment with the highest score is called *maximum segment pair (MSP)*. This is done for each *word-hit* produced by the previous step.

Evaluation

In this step all *MSPs* are tested for statistical significance. The score is converted into an *expectation value (e-value)*, which defines how many segment pairs with equal or higher score can be

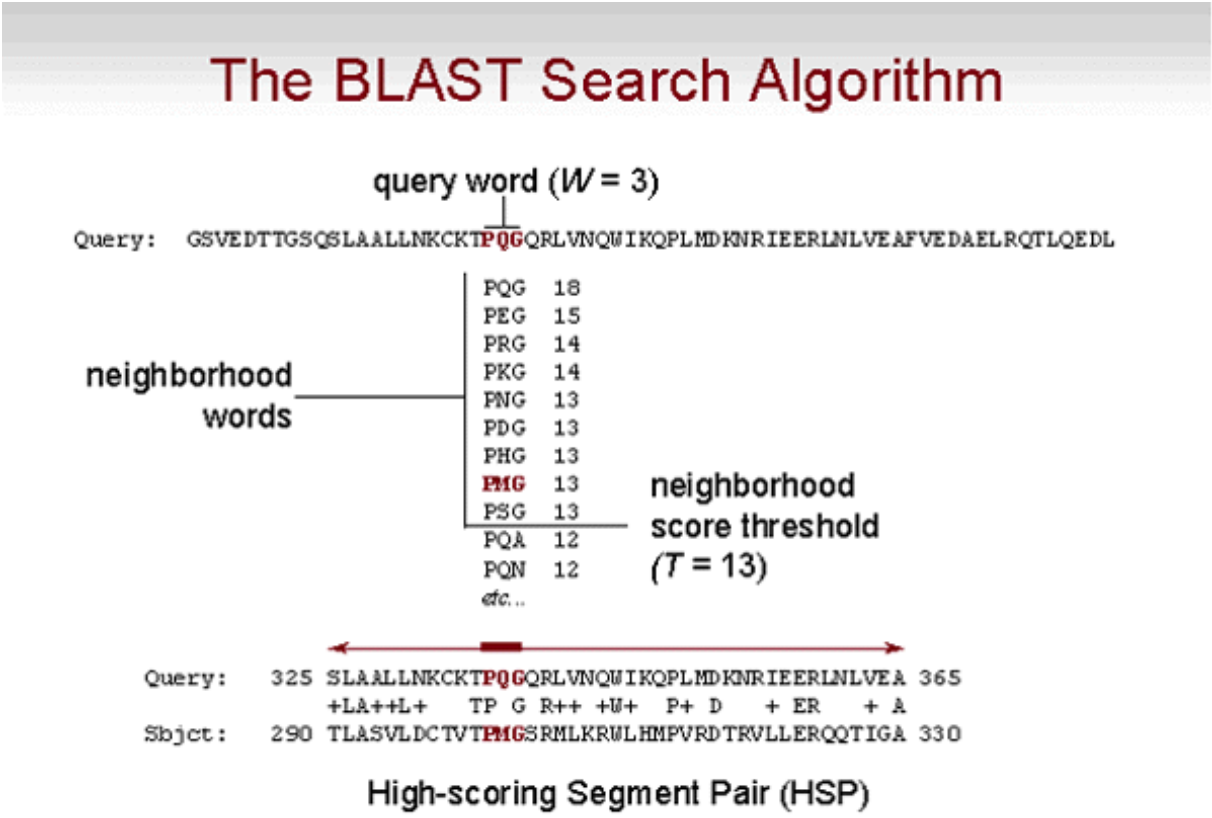

Figure 1: A simplified overview of the *BLAST* algorithm. Source: http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/BLAST_algorithm.html

expected by mere chance. Hence, a lower *e-value* indicates a more significant local alignment. If the *e-value* is below a defined threshold, the *MSP* is called *HSP* and is included in the output. In certain circumstances some of the *MSPs* are grouped and a common *e-value* is calculated.

2.3.2 Statistics

As a segment pair is an alignment of two short subsequences, it is no problem to calculate its score S with a given substitution matrix. For random sequences, the *e-value* for a segment pair with a score S greater or equal x is shown in Formula 1

$$E(S \geq x) = km'n'e^{-\lambda x} \tag{1}$$

To get the *e-value* for a known score S, Formula 1 is rearranged to Formula 2. This is also known as Karlin-Altschul equation [10].

$$E = km'n'e^{-\lambda S} \tag{2}$$

Below, the parameters of Formula 2 are explained. As already mentioned, E is the *e-value* and S is the Score. m' is the *effective length* of the query sequence and n is the *effective length* of the database. It is important to note, that these *effective lengths* differ from the *actual lengths*, which are simply the number of letters in the sequence or database. To avoid confusion, in this thesis effective lenghts are always written with prime (m' n') and actual lengths without (m, n). The differences between *actual lengths* and *effective lengths* are caused by edge effects, as it is not possible for an *HSP* to start on one of the last letters of sequence. k and λ are the Karlin-Altschul parameters. They depend on the used scoring matrix and on background frequencies of the used sequences. Without knowing these parameters, it doesn't make much sense to interpret the *raw score*, especially if results of different *BLAST* analysis are compared. Therefore, the *raw score* S is often converted into the *bit score* S', which is independent of k and λ (Formula 3)

$$S' = \frac{\lambda S - lnK}{ln(2)} \tag{3}$$

Substituting Formula (3) in Formula (2) leeds to Formula (4)

$$E = m'n'2^{-S'} \tag{4}$$

group statistics

When performing a *BLAST* search, it happens that multiple *HSPs* are found for the same pair of sequences. This is extra common for the search of a translated eucaryotic nucleotide sequence against a protein database. Keeping the biology in mind, this is not surprising. The *HSPs* represent the exons, and the regions between them are introns. As these exons may be part of the same gene, it is not allowed to consider the statistic of the corresponding *HSPs* separately. Therefore *BLAST* groups this *HSPs* and calculates a combined *e-value* for them [14]. There are two precondition for a set of *HSPs* that group statistics is applied, in all other cases the default statistics mentioned above is used.

- the *HSPs* must belong to the same sequence in the *BLAST database*

Program	Query	Database
BLASTn	Nucleotide	Nucleotide
BLASTp	Protein	Protein
BLASTx	Translated	Protein
tBLASTn	Protein	Translated
tBLASTx	Translated	Translated

Table 1: Traditional *BLAST* programs. Translated means that the sequence is nucleotide and translated into protein sequences, one for each of the six possible reading frames

- the *HSPs* must be consistently ordered. That means that the first *HSP* has to start on both sequences before the next *HSP*, and the *HSPs* do not overlap.

$$S'_{sum} = \lambda \sum_{i=1}^{r} S_r - \ln(kmn) - (r-1) * (\ln(k) + 2\ln(g)) - \log(r!) \tag{5}$$

Equation 5 shows how the *sum score* for a group of consistently ordered *HSPs* is calculated. For creating the *sum score* only the *hit sequence* instead of the whole *BLAST database* is used, therfore n is in this case only the length of the *hit sequence*. Corrections that consider the length of the whole *BLAST database* are performed in a later step. The parameter λ , k and m are the same as used in the equations for a single *HSP*. The number of consistently ordered *HSP* that are used are represented by r. The value g depends on the gaps between the single *HSPs*, so the closer the *HSPs* are together the higher the score is.
This *sum score* is used to create a common *e-value* for all involved *HSPs*. However, this conversion uses different formulas as the conversion for a single *HSP* known from Formula 2. They are not explained in more detail, as they are not relevant for this thesis. More information can be found in [10].

2.3.3 Programs

The two most popular implementations of the *BLAST* algorithm are NCBI-BLAST form the *National Center for Biology Information (NCBI)*, and WU-BLAST from Washington University. Unless otherwise noted, all statements in this thesis refer to the implementation of the *NCBI*. Depended on the type of sequences, different programs for the comparison have to be used. The five traditional ones are shown in Table 1. But also the type of problem is a criteria for the selection of the right program. For nucleotide sequences it is possible to use BLASTn as well as tBLASTx. With the first, the nucleotide sequences are directly compared, with the latter they are first translated to six protein sequences representing the six possible reading frames. This leads to different results, due to the degeneration of the genetic code. For instance the nucleotide *BLAST* will distinguish between CTT and CTC, on the other hand will the translated protein *BLAST* see both as the same amino acid leucine, at least in one reading frame.
There are also several programs that use slightly modified versions of the *BLAST* algorithm to solve specific problems. One approach to find distant homologies for protein sequences is PSI-Blast (position specific iteration Blast) [8]. It first makes a traditional BLASTp search and calculates a profile out of the hits. Then this profile is used for a further search, and the new hits are used to refine the profile. This is repeated until no new hits were found. For nearly exact

matches of nucleotide sequences, another variation called MegaBlast is the best choice, as it is much faster than a traditional BLASTn search. It uses by default a very large *word-size* and a greedy algorithm for the gapped alignment[15].

local *BLAST*

To perform a *BLAST* search, the *BLAST* package can be installed on a local machine [2]. This package includes the software for the *BLAST* search itself as well as several tools like formatdb, a program for creating own *BLAST databases*.

***BLAST* server**

The *NCBI* provides a physical server that allows everybody to run *BLAST* searches, the *NCBI www-blast Server*. It has several *BLAST databases* installed, but it is not possible to create and search against own databases. An easy way to access this server manually is to use its web-based interface [3]. For the automated access, there are two principal ways.

- A network client that allows remote TCP/IP connections. The *NCBI* provides such a client called Blastcl3.
- The BLAST URL api. The communication is directly encoded into HTTP requests using a standardized api [16].

The software running on the server is free, so it is possible to setup own servers.

2.3.4 Calculating Time

The time *BLAST* needs for a search depends on several issues, the most important are described in following:

- choosen program parameters.
- primary structure of *query sequence* and *BLAST database*.
- size of searchspace. The size of the searchspace is the number of theoretical startpoints for a local alignment. It is simply the product of the length of *BLAST database* and the length of the *query sequence*. Note, that again the the effective lengths (m' n') are used to consider edge effects. As mentioned before, *BLAST* is a heuristic method and does therefore not search the whole search space. Nevertheless its size is relevant for time a *BLAST* search needs: The longer the *query sequence*, the more words exist and the bigger the *neighborhood* of all words. The greater the neighborhood of all words, and the longer the database, the more *word-hits* are found and processed in the next steps. As shown in Figure 2, the time for a search against the same *BLAST database* is linearly dependent on the size of the *query sequence*.

[2]Executables for several platform as available on ftp://ftp.ncbi.nlm.nih.gov/blast/executables
[3]This interface can be found on http://www.ncbi.nlm.nih.gov/blast/Blast.cgi

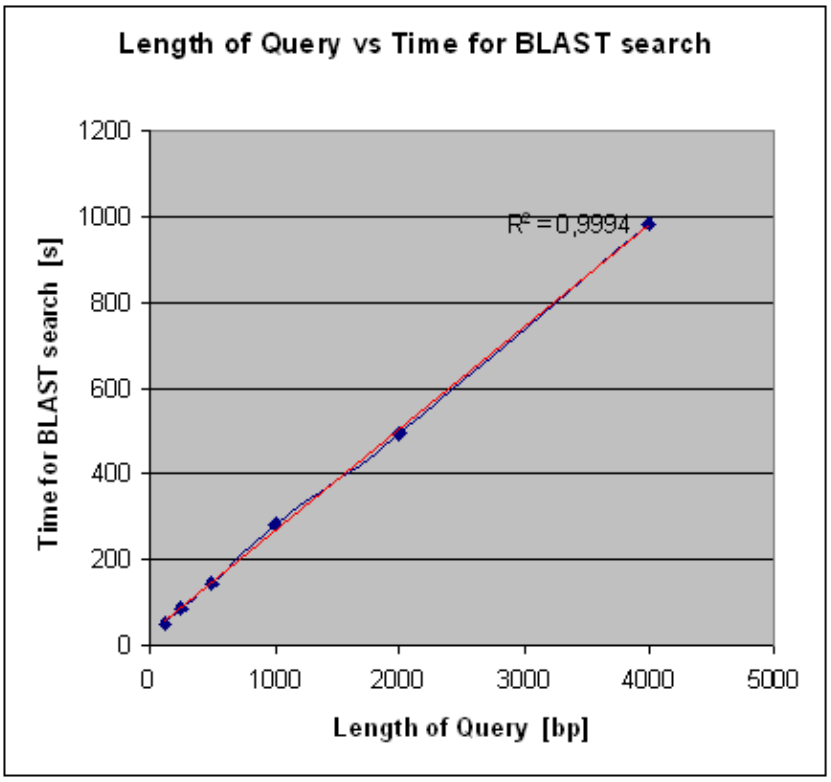

Figure 2: The length of random sequences vs the time needed for a *BLAST* search against *RefSeq Protein Database*. The blue squares represent the measured values. The red line shows a linear regression. The R^2 coefficient is 0,9994

2.3.5 Data Structure for a Result

To allow an easy parsing and postprocessing of *BLAST* results, the local *BLAST* as well an the *NCBI www-blast Server* can return the output formated in *eXtensible Markup Language (XML)* [17]. If the *BLAST query* contains more than one sequence, the *BLAST* outputs are simply printed consecutively without a common parent element. This is a violation of the W3C *XML* standard, as it claims a well formed *XML* file to have exactly one root element [18]. Depending on the used *XML* parser this leads to errors during the parsing process. Therefore, it is a good idea to add a root element manually. The typical structure of the tree is shown in Figure 3, in the following the main elements are described in more detail:

Blast Output

One of these element ist created for each *query sequence*. It contains fields for several information about the used program, database, *query sequence* as well as all used program parameters. Moreover it contains one or more elements of the type Blast Iteration.

Blast Iteration

In iterated versions of *BLAST* like PSI-*BLAST*, one of these elements is created for each iteration. However, an output for one of the traditional BLAST programs like BLASTn, tBLASTn, BLASTx or tBLASTx has only one iteration. It contains all the statistic information like the

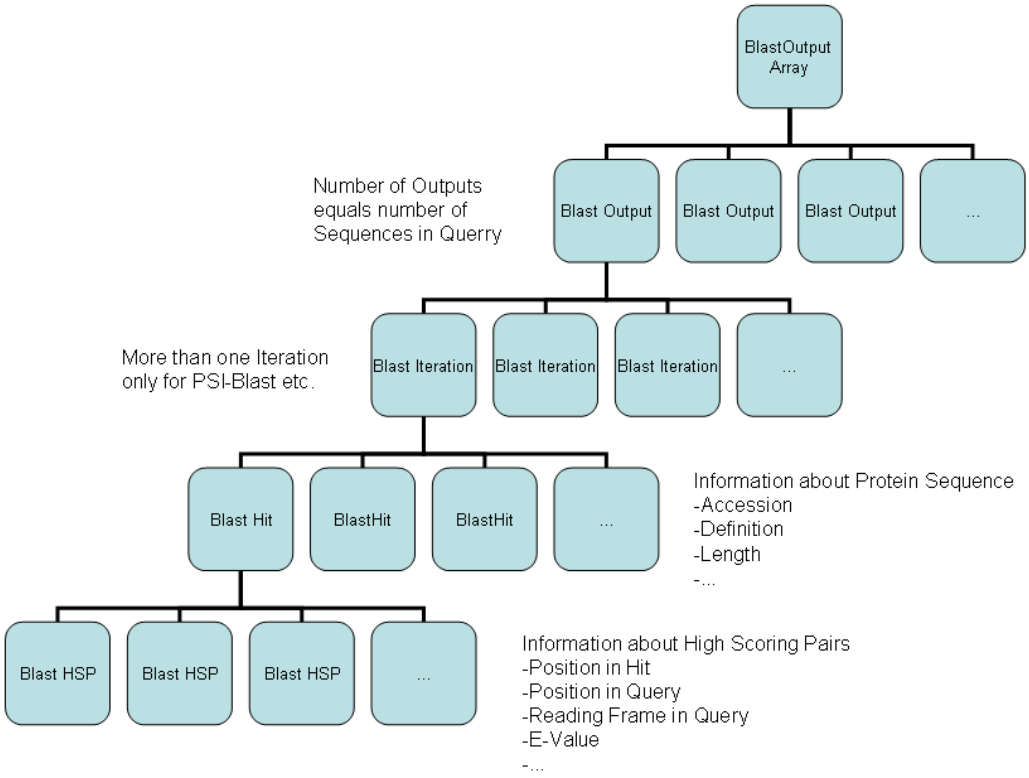

Figure 3: Structure of the output for *BLAST* search

effective search space or the values for the Karlin-Altschul parameters, and several Blast Hit elements.

Blast Hit

For each sequence in the database, where *BLAST* created significant alignments, an element of this type exists. It contains the most important information about the *hit sequence*, like the accession number, a short description and the length. Furthermore, it contains at least one Blast HSP element.

Blast HSP

All the *HSPs* are own elements. Each of them contains the corresponding alignment itself and additional information like the used regions on the sequences. Also the *expectation value* can be found there.

2.4 Reference Sequence Database

The *Reference Sequence Database (RefSeq Database)* [19] [20] provides a curated, non-redundant collection of nucleotide and protein sequences. It is built and distributed by the *NCBI*. In Release 26 (released at November 4, 2007) the protein part of the database contained more than 4

million records from more than 4500 species. As all entries are richly annotated, *RefSeq Protein Database* provides a solid reference for comparative annotation.

2.5 CLC bio

CLC bio is a company based in Aarhus, Denmark. Its business area is the whole field of bioinformatics, i.e. bioinformatics software, bioinformatics hardware as well as bioinformatics consulting. Founded in 2005, the company has now grown to more than 40 employees.

2.5.1 Workbenches

CLC bio provides several software applications, the *CLC bio Workbenchs*, for sequence analysis. Beside a free version with basic features, there are also specialized commercial versions for the analysis of DNA, RNA and protein data, and a combined version that includes all the features. As the used programming language is Java, the software runs on Windows, Linux as well as on Mac OS. Figure 4 shows a screenshot of the Combined Workbench.

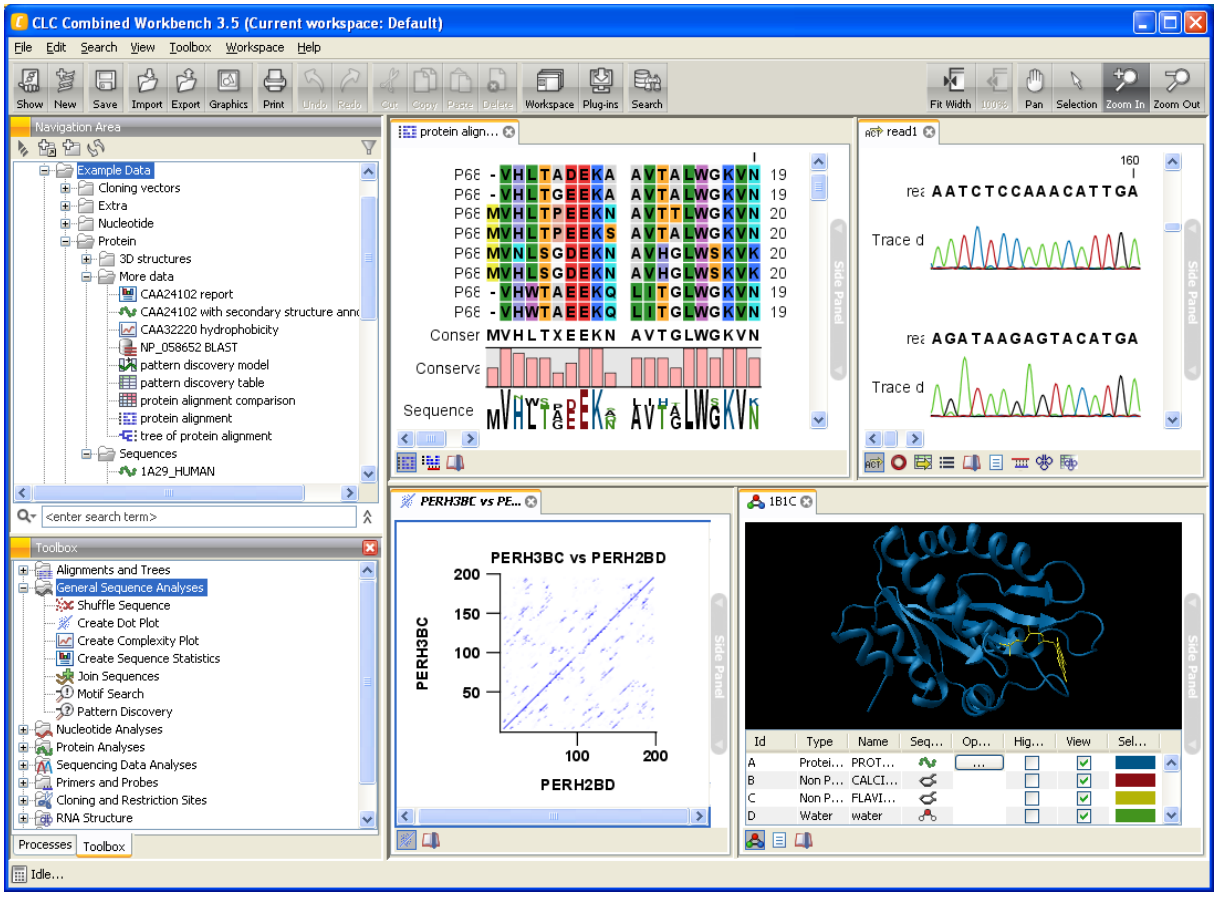

Figure 4: Screenshot of the *CLC bio Workbench*.

2.5.2 Software Developer Kit

To enable the users to customize the software, *CLC bio* released a *Software Developer Kit (SDK)* in 2007. It provides a powerful *Application Programming Interface (API)*, which includes many useful algorithms and datatypes for bioinformatics, as well as the functionality to interact

with the workbenches. Software created with the *SDK* is not standalone but runs as plugin in one of the commercial workbenches. Each plugin can add one type of functionality, like a new editor for an existing datatype, a parser for the import of data in a new format or an action that modifies objects. For more comprehensive projects multiple plugins can be grouped to build a module.

Limits of the SDK

In the actual version 1.0 of the *SDK* it is not possible to create own datatypes. Therefore, plugins have to use datatypes that are already defined.

2.6 Task Definition

The aim of the project is to create a software module for the annotation of newly sequenced *DNA* sequences. The main method on which the annotation process should be based is a BLASTx search against the *RefSeq Protein Database*. The search itself should runs on the *NCBI www-blast Server*. Annotations should be transfered from the *hit sequences* to the unknown sequence. Megabase-sized sequences should be handled. As this approach can only be a first step in an annotation process, the possibility to extend the module with further methods should be given. Moreover, it is important that full control of the annotation process is applied to the user. The software should be a module for the *CLC bio Workbench*.

3 Handling of Megabase-Sized Sequences with BLAST

3.1 Problem

A *BLAST* search of a megabase-sized *DNA* sequence against *RefSeq Protein Database* is very calculation intensive. Therefore it is desirable to introduce some kind of parallelization.

3.2 Different Approaches

To solve this problem, several approaches are thinkable, but all of them have their advantages and disadvantages. The goal of all approaches is to return a *BLAST* result that is as similar as possible to the result of a the *BLAST* search with the *DNA* sequence as *query sequence* and the *RefSeq Protein Database* as *BLAST database*. To make the algorithm parallel, the *BLAST* search is split in several smaller subsearches. The results are merged, and the statistics are corrected. A schematic overview of the differnt approaches is shown in Figure 5.

3.2.1 Swap Database and Query

In a preprocessing step, *BLAST* converts the database into a *BLAST database*, which is in a special format that enables faster searching. Therefore, it is common to use the longer sequence (or list of sequences) as *BLAST database* and the shorter one as *query sequence*. As the *RefSeq Protein Database* and the unknown sequence are both very long, it is not obvious which one should be used as query and which one as database. Using *RefSeq Protein Database* as *BLAST query* has the advantage, that it already consists of many sequences, which are much shorter than the unknown sequence. Therefore, a separate search for each sequence can be performed. So the possibility for running the search parallel is given as shown in Figure 5 B. The complexity for each subsearch is defined by the size of the sequences in *RefSeq Protein Database*. As in each search the whole *DNA* sequence is used, there is no problem regarding group statistics.

3.2.2 Split Data

In the following approaches, *RefSeq Protein Database* is used as *BLAST database* and the unknown sequnence as *query sequence*. To split the search in several less complex searches there are two possibilities, either the database or the unknown sequence is split.

Split Database

The *RefSeq Protein Database* contains many sequences. Therefore, it is no problem to split it in several smaller databases. As shown in Figure 5 C a *BLAST* search is performed for each of this subdatabases as *BLAST database* and the unknown sequence as *query sequence*. The complexity of each subsearch can be adjusted by the size of the parts of the database. Like in the previous approach there is also no problem regarding group statistics, as in each search the whole *query sequence* is used. In most existing implementations that parallelize a *BLAST* search this approach is used [21][22].

Split Query

The second possibility is to split the unknown sequence as shown in Figure 5 D. A *BLAST* search is performed for each of this subsequences as *query sequence* and the *RefSeq Protein Database* is used as *BLAST database*. To avoid loss of significant hits at the cutting sites, it is necessary to give the subsequences an overlap [23]. As the sequence is split, there is however

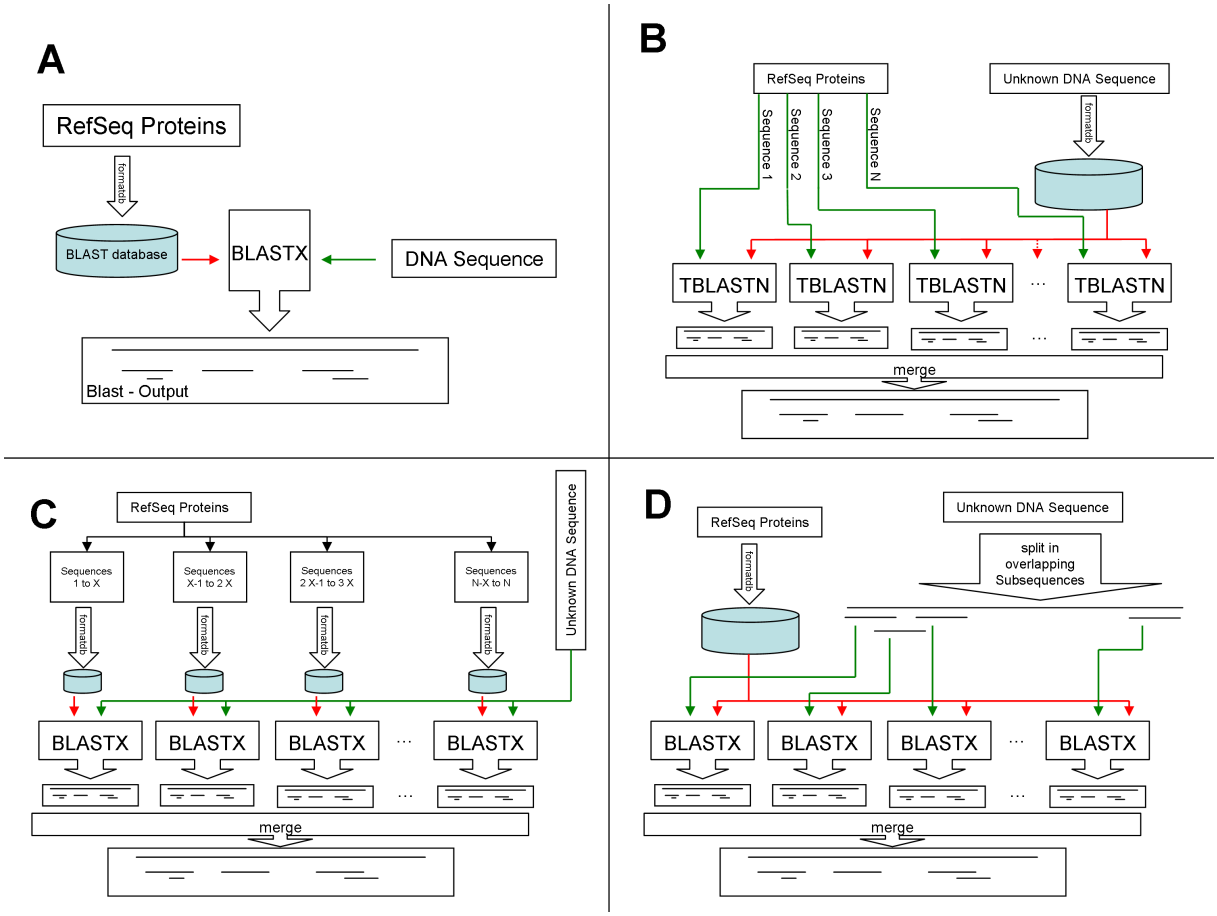

Figure 5: Schematic overview of different approaches for a *BLAST* search for a nucleotide sequence against *RefSeq Protein Database*. A green arrow indicates that the input is used as *query sequence*, red arrows stand for *BLAST database* **A: (Default)** A BLASTx search is performed. As there is only one *BLAST* search performed, the calculation can not run parallel. **B: (Swap Database and Query)** The nucleotide sequence is used as *BLAST database*. For each sequence in *RefSeq Protein Database* a tBLASTn search is performed, and the results are merged. **C: (Split Database)** The database is split in smaller subdatabases. A BLASTx search is performed against each of them, and the results are merged. **D: (Split Query)** The nucleotide sequence is split in overlapping subsequences. For each of them a BLASTx search against *RefSeq Protein Database* is performed and the results are merged.

	Swap Query and Database	**Split Database**	**Split Query**
used program	tBLASTn	BLASTx	BLASTx
complexity of each subsearch can be adjusted	no	yes	yes
recalculate statistics	simple	simple	complex (group statistics)
runs on *NCBI www-blast Server*	no	no	yes

Table 2: comparison of different approaches for splitting a *BLAST* search between the *RefSeq Protein Database* and a long sequence. In all cases the search is split in subsearches

the possibility that the recalculation of the group statistics may cause problems. The complexity of each subsearch is defined by the selected size of the subsequences.

3.2.3 Conclusion

All approaches solve the problem as they split the search in less complex subsearches. As short overview is shoen in Table 2. Unlike the other appraoches, the last one is the only one that uses the *RefSeq Protein Database* as *BLAST database* without modification. However, for the search on a *NCBI www-blast Server* only some predefined databases can be used as *BLAST database* and the *RefSeq Protein Database* is one of them. As using the *NCBI www-blast Server* is a part of the task definition for the project, only the split query approach is feasible.
This also solves a further problem regarding the *NCBI www-blast Server*: Even if there is no theoretical upper limit in *BLAST*, neither for the query sequence nor for the database, there are practical limits when using the *NCBI www-blast Server* as it limits the cpu-time per query. As mentioned in Section 2.3.4 the usage of cpu-time of a *BLAST* search depends not only on the size of the *query sequence* and *BLAST database*, but also on some inner attributes of the sequences as well as on the used parameters. But to get a rough picture on the limit of what *NCBI www-blast Server* allows, some BLASTx searches against *RefSeq Protein Database* were performed using random nucleotide sequences [4] and all parameters default. The search completed without any problems for sequences with 50*Kilo base pairs (kbp)*, but sequences with 100*kbp* resulted in an error message from the server.

3.3 Split Query

A schematic overview for a *BLAST* search using overlapping subsequences was already shown in Figure 5 D. In the following this approach is explained in more detail.

3.3.1 Overlapping Subsequences

For the creation of overlapping subsequences, there are two main parameters. One is the size of a subsequence. Choosing a too small value for it will result in several problems: For a large *HSP*, it may happen that it consists of more residues than a subsequence. So, *BLAST*

[4] All random sequences mentioned in this thesis have a GC content of 0.5

cannot extend the alignment and will only return a truncated *HSP* with an *e-value* too high. Furthermore, a small subsequence size leads to a huge overhead, as the amount of subsequences is reciprocally proportional to their size. But too large subsequences are also undesirable, as they cause problems with the length restriction of the *NCBI www-blast Server*.
The second parameter is the number of bases two subsequent subsequences share, that means the overlap between them. A very small overlap, or as an extreme case no overlap at all, decrease the probability for an *HSP* to occur on a single subsequence without being split. On the other hand, a very large overlap causes needless redundant calculations, as many parts of the originals sequence are used twice. Therefore a badly adjusted size of the overlap can have negative effects on time and result of the search.
In order to enable a later reassembling, each result has to know the offset of the corresponding subsequence. This is simply the number of bases in the original sequence before the first base of the subsequence.

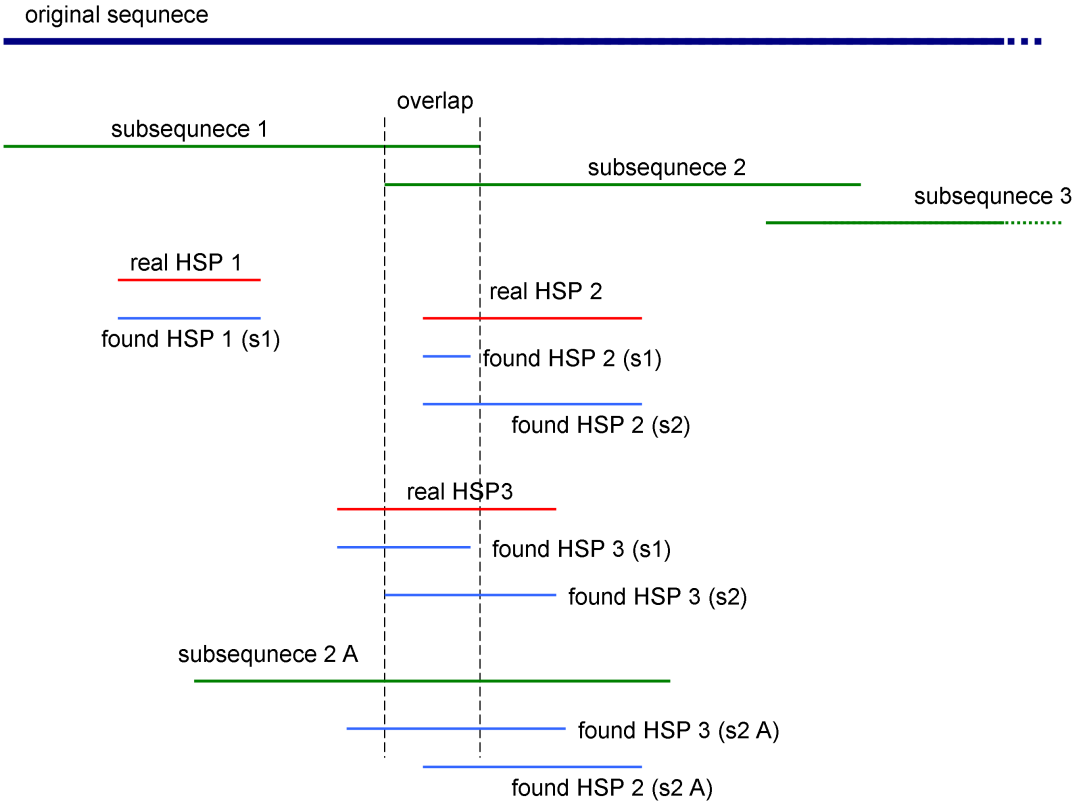

Figure 6: An long sequence (dark blue) is split in overlapping subsequneces (green). The red lines indicates *HSPs* a conventional *BLAST* search without splitting would had found. The light blue lines show the *HSPs* found in a *BLAST* search for each subsequence. Note that HSP 3 is in none subsearch found completely. Therefore a additional subsequence 2 A is created, and the HSP 3 is completely covered. This Figure shows also, that in this approach many duplicate (HSP2s2 and HSP2s2A) and truncated (HSP2s1, HSP3s1, HSP3s2, HSP2s2A) *HSPs* are created that have to be eliminated in the merging step

3.3.2 HSPs in the Overlap

To reduce redundant calculation, the size of the overlap is limited. A real *HSP* that occurs on the overlap but exceeds the overlap on both sides will therefore be trimmed. As this goes along with a loss of information, such cases have to be recognized. Therefore, an additional *BLAST* search is performed as shown in Figure 6. Obviously, all found *HSP* that fill the whole overlap are relevant, but these are not the only ones. In the step when *BLAST* extends the *word-hits*, the score does not increase steadily but has local maxima and minima. If such a local minimum is exactly at the end of the overlap, *BLAST* will reduce the alignment to the point of the last local maximum score. So, the resulting *HSP* will not fill the whole overlap, but is nevertheless wrongfully trimmed.

3.3.3 Merge Results

The merging process consists of following steps:

- **correction of parameters.**
  To get the start and end of the *HSP* on the original sequence, the offset of the subsequence has to be added. Also the reading frame depends on the offset of the subsequences and has therefore be corrected.

- **correction of statistics.**
  With the given *effective lengths* for the subsequences and the complete sequence, the statistcs for single *HSPs* can easily be corrected using the formulas shown in Section 2.3.2. However, the group statistics can not be recalculated in such an easy way as it is possible that some *HSPs* that are only significant in the context of other *HSPs* are separated from their groups when the sequence was split. Furthermore, there are problems with the statistics section in the *XML* format as shown in the appendix. Therefore these corrections are not performed.

- **delete duplicate *HSPs*.**
  Due to the overlapping sequences, some *HSP* may occur twice, and therefore one of them has to be deleted. In Figure 6 HSP2s2 and HSP2s2A are duplicates.

- **delete truncated *HSPs*.**
  As the hit can include the edge of the overlap, a *HSP* can be truncated. Such an *HSP* can be identified as it is always a sub*HSP* of a complete *HSP*. A *HSP* is defined as follows: It does not start before or end after the complete *HSP*. Furthermore, is it necessary that it is in the same reading frame of the *query sequence* and refers to same sequence in the *BLAST database*. In Figure 6 HSP2s1 is a sub-*HSP* of HSP2s2. HSP3s1 and HSP3s2 are sub-*HSPs* of HSP3s2A.

3.3.4 Parallel Requests

NCBI www-blast Server allows multiple requests. However, it gives users that overload it a lower priority. Therefore it is necessary to avoid too many parallel requests. This can be done by keeping only a defined number on requests alive and start a new one not until an old one has finished. Furthermore, there has to be a minimum time of a few seconds between the requests.

x

4 Transfer of Annotations

The results produced with *BLAST* represent the similarity between the aligned regions in the *query sequence* and the *hit sequence*. Even if similarity is not sufficient to ensure homology, it is nevertheless a strong indication for it. As a result, it is very likely for similar sequence to have similar biological properties. Newly sequenced sequences obviously don't have information about these properties. For sequences in databases like the *Reference Sequence Database* however, many such properties are stored as *annotations*. Therefore, an approach for assigning *annotations* to an unknown sequence is to perform a *BLAST* search against such a database and transfer the annotations of the well-annotated *hit sequencees* to the *query sequence* depending on the local alignment in the *BLAST* result. Figure 7 shows the basic approach. As the *BLAST database* contains usually many sequences, it is likely that for some regions on the *query sequence* many local alignments are created.

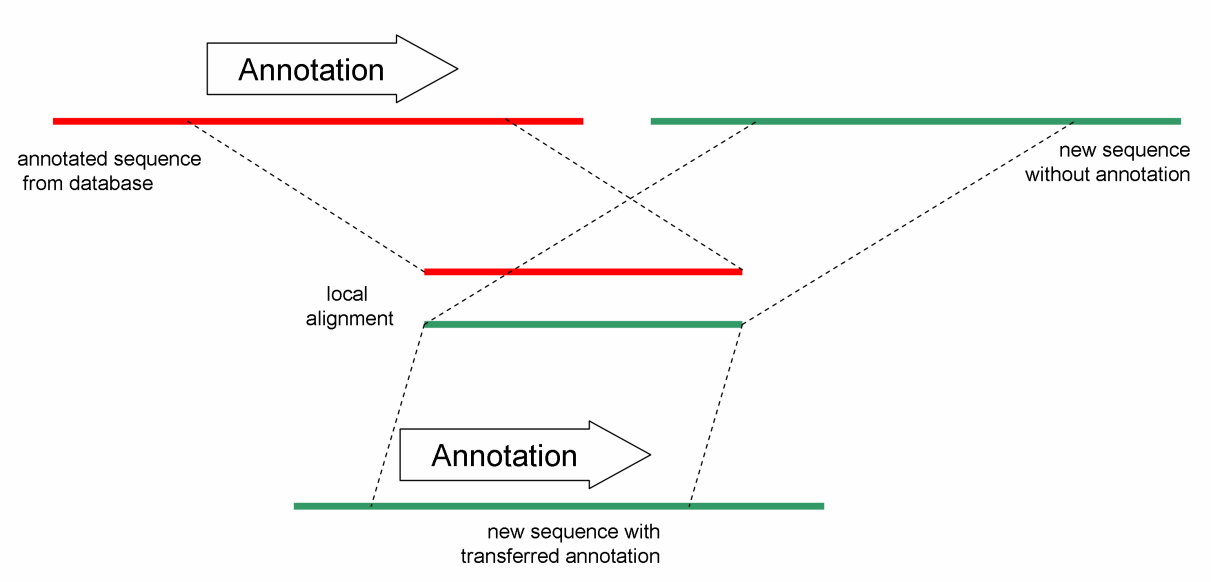

Figure 7: Basic approach for the transfer of an annotation from a well annotated sequence to a homologous sequence without annotations. Note that the region that is used for the local alignment covers the whole annotation.

Not all annotations of the *hit sequencees* are transfered to the *query sequence*. So annotations that cover exclusively regions on the *hit sequence* that are not part of the local alignment are ignored. Furthermore, in some cases the transfer of annotations in regions that are completely covered is also not desired. Especially for regions of the *query sequence*, where many local alignments to different sequences in the database were created, as this can easily lead to an annotation overload. Therefore, a filter system for the annotations is introduced that covers following parameters:

- **Type of annotation.**
  Only annotations of a defined list of types are used.

- **Significance of the similarity.**
  Annotations from an *HSP* with an *e-value* above a threshold are filtered.
- **Species of the *hit sequence*.**
  This option enables to use only annotations from sequences that belong to a defined group of species.

Partly covered annotations

So far, only two cases were considered. The annotation was either outside the region for the alignment and therefore ignored, or it was completely covered by this region. But it is also possible that the alignment is only partly covered. Figure 8 gives an overview of how this situation can be handled. There are three principal ways. First, the whole annotation is transfered anyway. (Figure 8A) This may be the best choice if only a minor part is not covered by the alignment. Second, the annotation is completely ignored, what seems to be most reasonable if only some residues are part of both, the alignment and the annotation. (Figure 8B) The third possibility is to transfer only the part of the annotation that is actual part of the alignment, and mark the truncated end as uncertain. (Figure 8B)

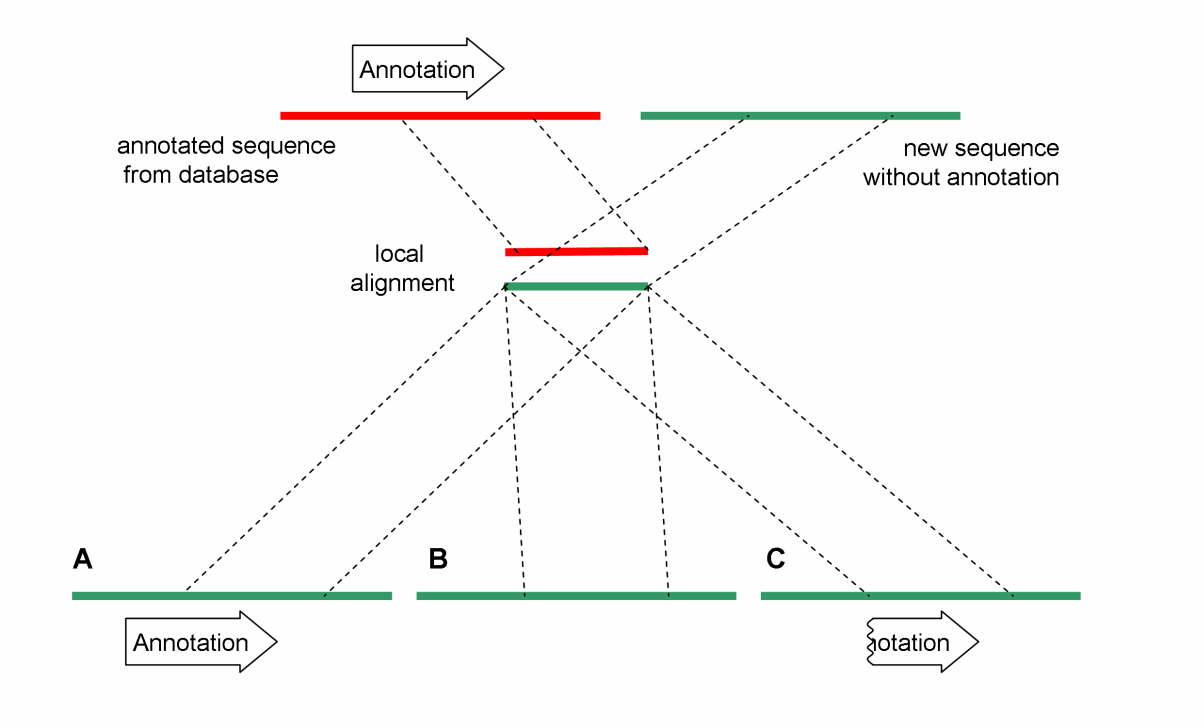

Figure 8: Three options for the transfer of an annotation that is only partly covered by the alignment. **A:** The whole annotation is transferred. B: The annotation is discarded. C: The annotation is truncated and only the part that is covered by the alignment is transferred. Note that it has no meaning whether an annotation is written on top or below a sequence

Split annotations

The previous examples concerned each local alignment separately. However, a *BLAST* search can produce several *HSPs* for the same pair of sequences, and their regions on the *hit sequence*

can both cover the same annotation. As mentioned in Section 2.3.2 this is very common for a BLASTx search of an eucaryotic DNA sequence against a protein database due to inron - exon structure. Figure 9 shows an example of such a situation.

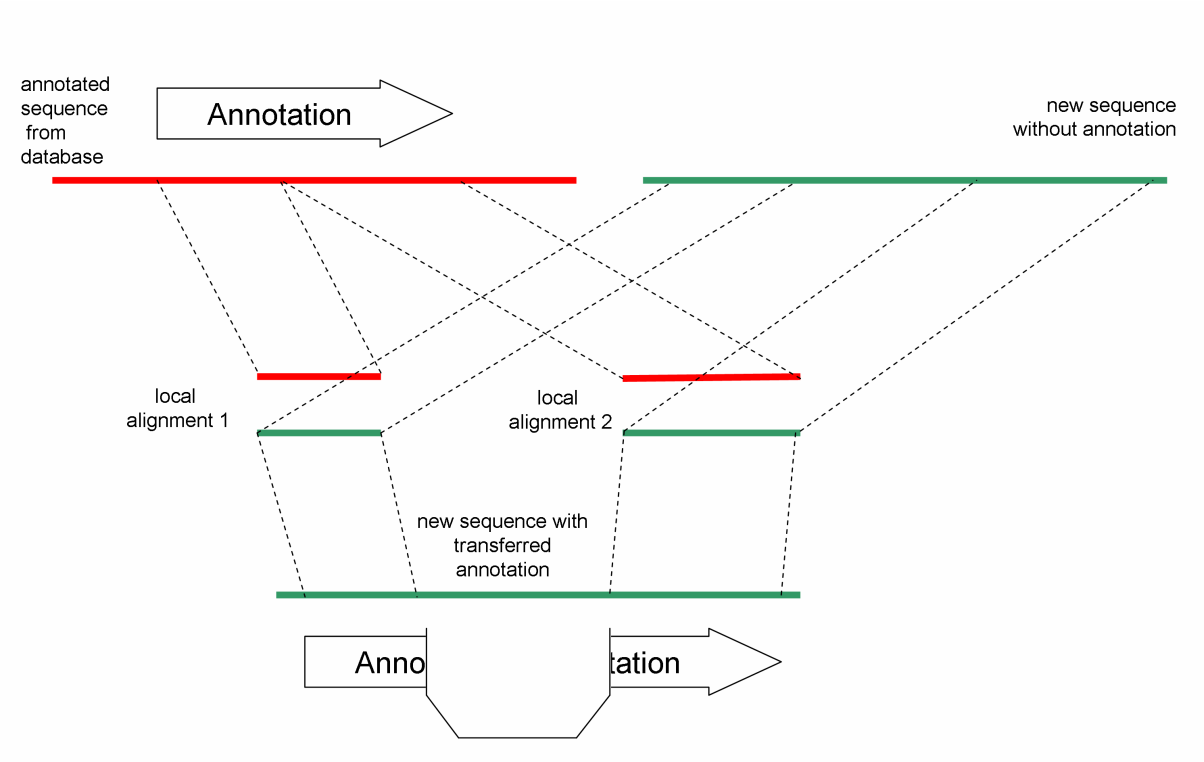

Figure 9: Transfer of an annotation that covers two local alignments. The parts of the annotation are not transferred separately but as one annotation with an inserted gap.

5 Design and Implementation

5.1 Product Specification

5.1.1 Aim

The aim of the project is to create a software module for semi-automated annotation of megabase-sized *DNA* sequences by homology search. The homology search is based on a *BLAST* search on an external server.

5.1.2 Prospective Users

The software module is aimed toward researchers that work with newly sequenced *DNA* sequences. The function of this sequences is not known yet, and therefore he wants to get a first overview of regions of interest and possible functions of this regions.

5.1.3 Functional Requirements

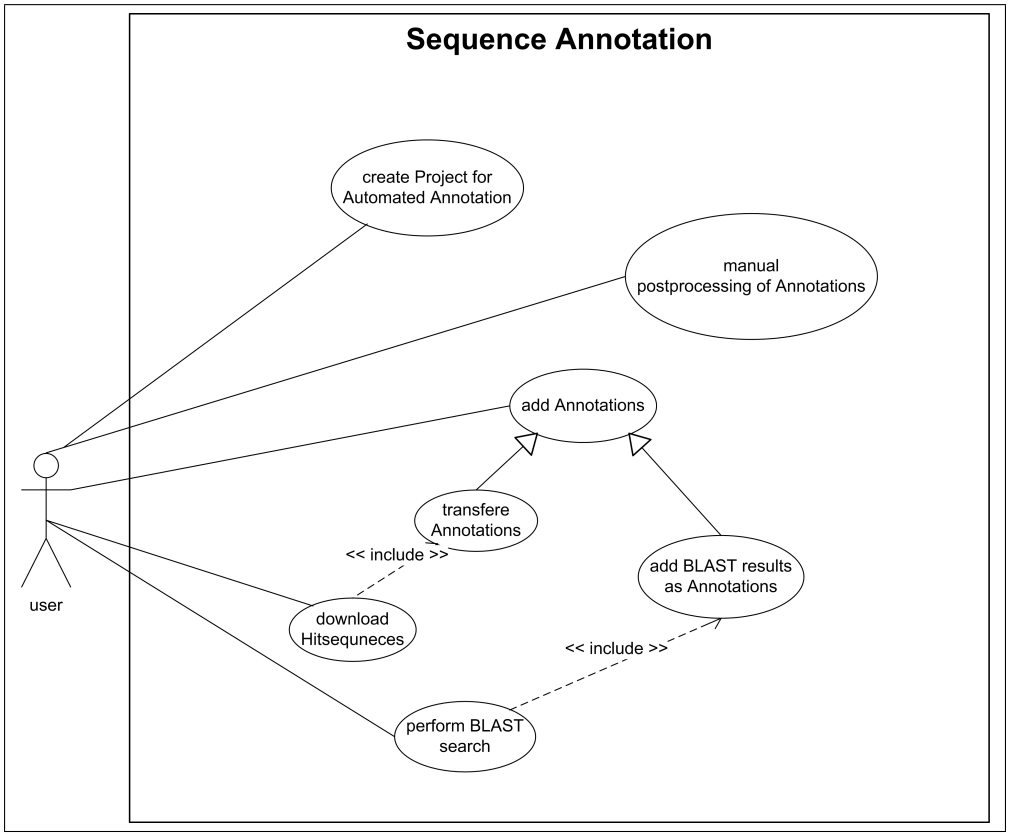

Figure 10: Use Case Diagram for Sequence Annotation.

Must criteria

The following functionalities have to be implemented. A use case diagram is shown in Figure 10

- The user has a *DNA* sequence and can create a new project for it.

- The user can perform a BLASTx search against *RefSeq Protein Database* on an *NCBI www-blast Server*. Size limitation by restrictions of the *NCBI www-blast Server* must be handled.
- The system has to provide the necessary tools for splitting the *BLAST* search in smaller subsearches and submit them as parallel requests to the *NCBI www-blast Server*.
- To avoid a overload of the *NCBI www-blast Server*, the system has to take care of a reasonable number of parallel requests.
- For an performed *BLAST* search, the user can download all *hit sequences*
- The result of a *BLAST* search can be added as annotation to the *query sequence*
- Annotations of the *hit sequences* can be transferred to the *query sequence*
- The system provides tools for a manual postprocessing of the annotations

Can criteria

These functionalities are nice to have, but not necessary.

- To prepare the extension of the module with further methods for annotation, it can provide the appropriate interfaces.
- The module can provide its logic for parallel requests and server load, to use it for sending *BLAST* requests from a queue of many relatively short sequences.
- As an option, the system uses only the parts of the sequence inside an *ORF*

Out of scope

To limit the scope of the project, following functionalities are not part of the systems.

- The system can not collaborate with a local *BLAST* installation.

5.1.4 Non Functional Requirements

- The system has to keep the user informed about the running processes. Therefore, the system provides the progress measured in percentage, as well as a textual description of the current operation.
- The module has to use the same look and feel as the *CLC bio Workbenches*
- The software has to be cross-platform.
- The users must always be able to pause running processes and continue them later.

5.1.5 Development Environment

As far as possible, the software has to be implemented using the *SDK* of *CLC bio*. The parts of the software that can for technical reasons not be realized with the *SDK*, are directly added to the *CLC bio Workbench*. This is mainly the creation of new datatypes. As the *SDK* is written in Java 1.4, the module has to be written in the same programming language.

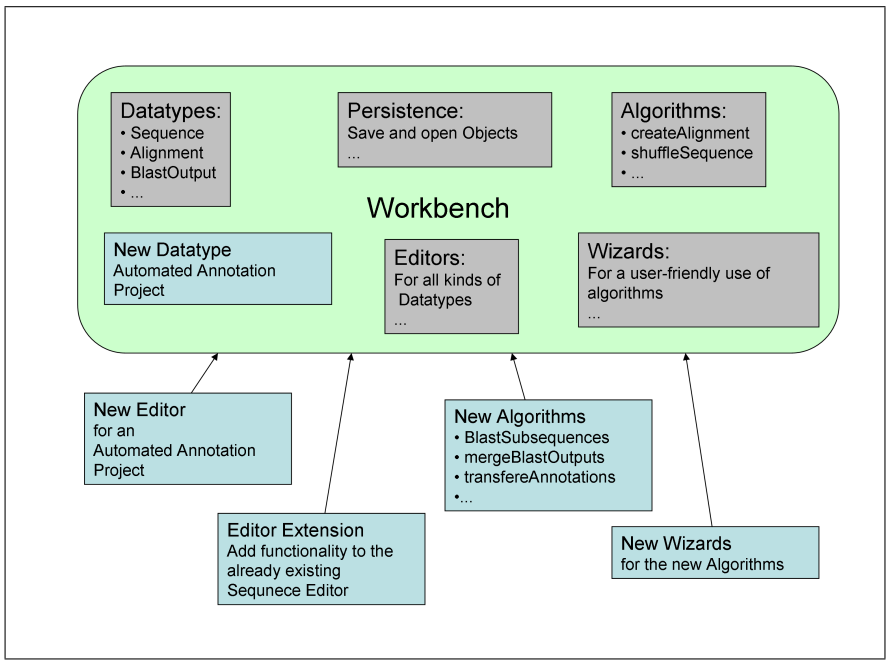

Figure 11: Schematic overview of the components for this project. **Green box:** This is the workbench itself. **Gray boxes:** These components did already exist. **Blue boxes:** These components are part of this project. The arrows indicate that the components in the blue boxes are plugins for the workbench. Note that the new datatype is not a plugin but part of the workbench itself.

5.2 Software Architecture

As shown in Figure 11 the project consists mainly of several smaller plugins for the *CLC bio Workbench* and a new datatype. In this section all these parts are explained in more detail.

5.2.1 Automated Annotation Object Datatype

Datatypes play a special role in the *CLC bio Workbench*, as instances of them are the only objects that can use the persistence framework. It is obvious that it is desirable to have the possibility to save and restore the results of calculations. Therefore, there are two possibilities when creating new algorithms. First, the result type of this algorithm is of an already existence datatype. This is for instance the case for an algorithm that shuffles a sequence. But this is not possible for algorithms that create results that do not fit any existing datatype. Therefore one of the major tasks of the AUTOMATED ANNOTATION PROJECT DATATYPE is to control the persistent data. These data are:

- the original *query sequence*
- a *BLAST* result for each created subsequence
- the downloaded *hit sequences*
- the annotated clones of the *query sequence*

Most of these data are standard in bioinformatics, like SEQUENCES and BLASTOUTPUTS. Therefore the default datatypes provided by the *CLC bio Software Developer Kit* can be used.

As shown in Figure 12, the AUTOMATED ANNOTATION OBJECT does only contain references to this objects, together with some additional information. For a BLASTOUTPUT, this additional information contains for instance the offset of the used subsequence and boolean flag that indicates whether this BLASTOUTPUT is empty. This flag is necessary as in a simple *BLAST* search no BLASTOUTPUT is created if no hits were found and so this information would be lost.

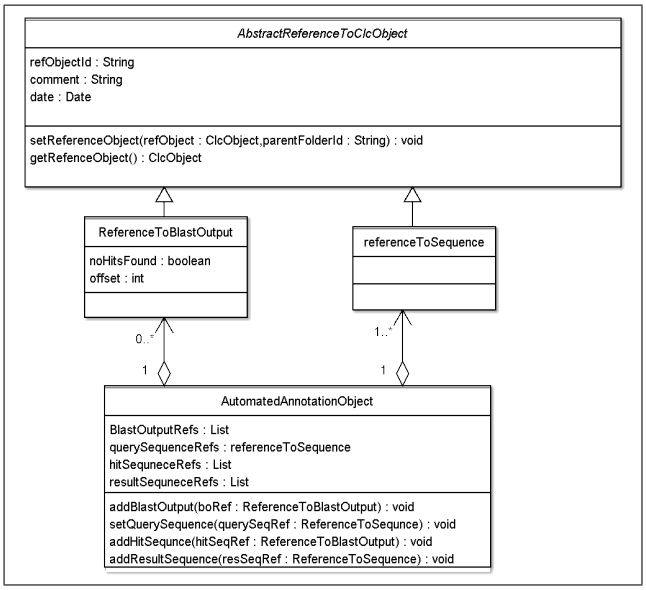

Figure 12: Classdiagram of the AUTOMATED ANNOTATION OBJECT and some classes that enable an easy handling of references to CLCOBJECTS. Note that for the sake of clarity this diagram is simplified.

5.2.2 Automated Annotation Object Editor

To open the an object in the *CLC bio Workbench*, there must be a corresponding editor. Obviously, the main goal of an editor is to display the object, but it can also be the start point for manipulations on this object. This is the case for the AUTOMATED ANNOTATION OBJECT EDITOR, as the start of the *BLAST* search, the download of the *hit sequencees* and the annotation process itself are initialized within this view. A screenshot of this editor is shown in Figure 13.

5.2.3 Create a new Automated Annotation Object

For the creation of a new AUTOMATED ANNOTATION OBJECT, the *DNA* sequence has to be given as argument. Furthermore, a folder in the persistence must be specified, where data associated with this object should be stored. Figure 14 shows the structure created for this object.

5.2.4 Blast of Subsequences

The *BLAST* search for subsequences consists mainly of two basic challenges. First, subsequences have to be created and depending on the results of the searches also shifted ones. Second, the requests have to be sent in a way that does not overload the server.

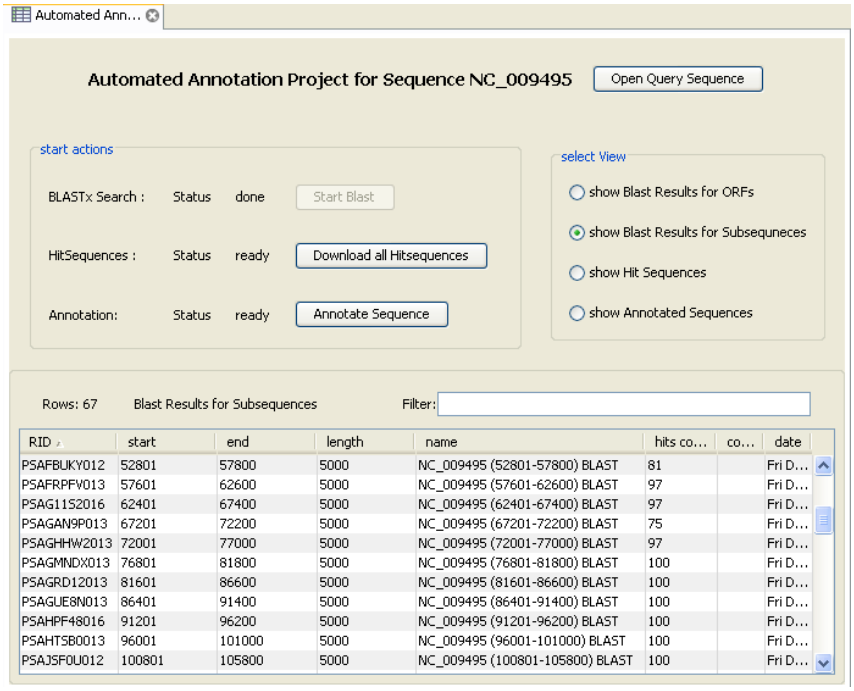

Figure 13: An AUTOMATED ANNOTAION OBJECT opened in its corresponding Editor.

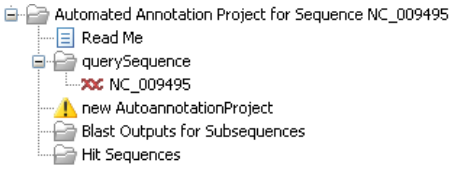

Figure 14: Objects and folder structure created for an AUTOMATED ANNOTATION OBJECT

Create subsequences

Basically, the algorithm takes a long sequence and creates shorter subsequences with a defined overlap. However, there is more to it than that. As already described in Section 3.3.2 it is problematic if a hit exceeds the overlap on both ends. In this case, a further subsequence is created which is shifted by the half length of a subsequence. So the critical region is in the middle and the new *BLAST* search covers the whole hit. A further exception is made for the last subsequence: As the size of the subsequence has an impact on the calculated *expectation value*, it is important that all subsequences are of the same size to keep the results comparable . Therefore, the overlap between the last two subsequences can vary. A flowchart is shown in Figure 15.

Send BLAST requests

The used *API* provides the functionality to send a single request to the *NCBI www-blast Server* and parse the result to an object for the workbench. But to benefit from the fact that the *BLAST* search is split into smaller subsearches, these searches have to be performed parallel. However, on the *NCBI www-blast Server* server the number of requests a user can run parallel is limited. Therefore, not all requests are sent on the same time, but the number of running searches is

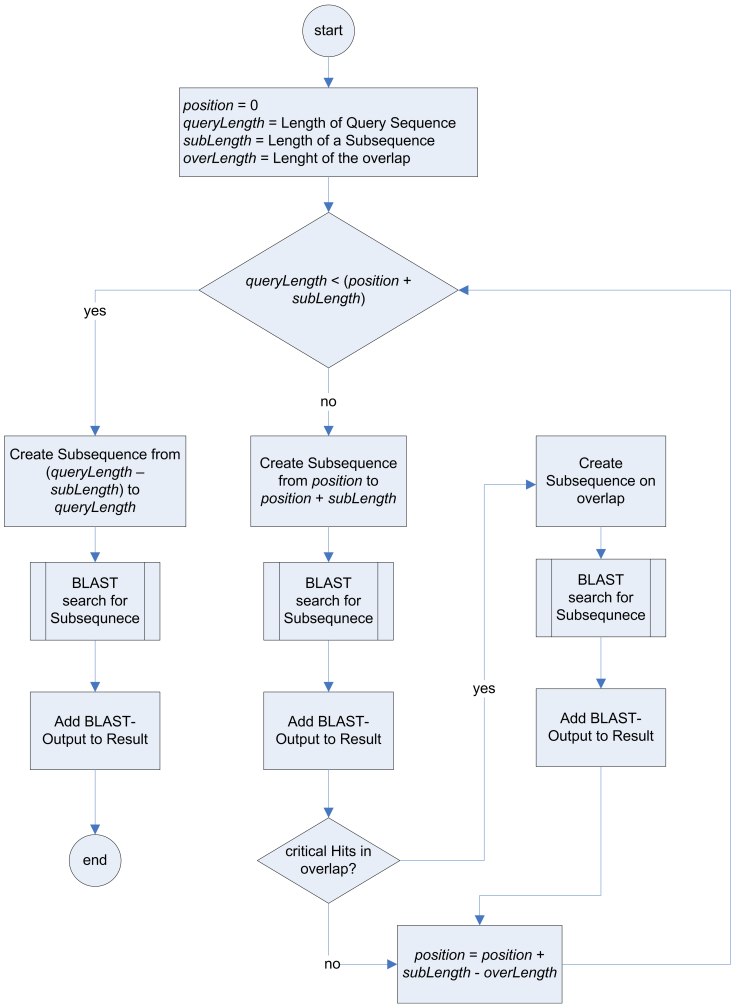

Figure 15: Flowchart for the creation of the overlapping subsequences. For the sake of clarity this flowchart is slightly simplified, so it does not show how the BLAST searches a parallelized.

kept constant. A petri net that solves this problem is shown in Figure 16. To enable further processing, these results are stored together with its offset, i.e. the number of bases previous to the corresponding subsequence.

Variation

In a variation not overlapping subsequences but the regions annotated as *Open Reading Frames* are used for *BLAST* searches. Therefore, abstract classes are used for managing the parallel requests and the server load. The difference are defined in specialized classes for both approaches. This is mainly the handling for the results and the creation of new sequences. For a class diagram see Figure 17.

5.2.5 Annotate with Blast Results

Merging all the BLASTOUTPUTs for each subsequence to one BLASTOUTPUT has a disadvantage. The resulting object would be very big and the BLASTOUTPUT object is designed for comparable short sequences. Therefore, a slightly other approach is made. The merged *HSPs*

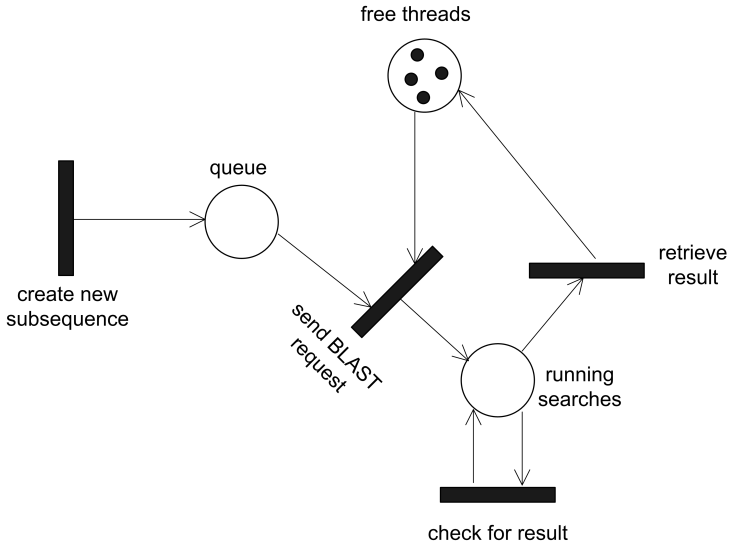

Figure 16: Petri net for keeping the number of active requests constant. In this case 4 searches can run on the same time. As recently as one of this searches is finished a new request is sent.

are add as annotations to the unknown sequence. Also information about the *BLAST* search itself is add as annotation. Reference to all original BLASTOUTPUT are kept, so they can easily be opened in the default editor. See Figure 18

5.2.6 Download of Hitsequences

In order to transfer annotations from the *hit sequences*, they have to be downloaded. As only the annotations are needed, it would be sufficient to download the sequence without its primary structure, but it seems that the common databases don't support this. For the download of a sequence from *NCBI* and the parsing into the workbench there are methods available in the used *API* and can therefore be used as a black box.

5.2.7 Transfer of Annotations

An iteration is made over each annotation on every *hit sequence*. If the annotation fits the criteria set by the user, it is transfered according to the alignment created by the *BLAST* search.

5.2.8 Sequence Editor Extension

To the default sequence editor of the workbench, functionality is added. This is mainly filtering of shown annotations by the *e-value* of the corresponding *HSP*, and the possibility to open the appropriate BLASTOUTPUT directly from this view. A graph gives an overview how the annotations are distributed on the sequence (Figure 18.

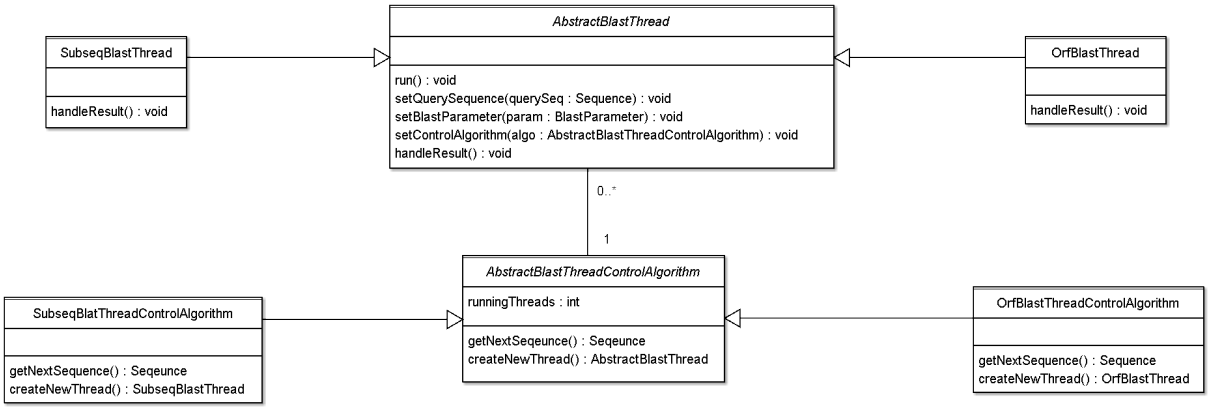

Figure 17: Class diagram for sending parallel requests for different set of sequences. **Center**: The abstract classes. For each *BLAST* search a ABSTRACTBLASTTHREAD is created. The ABSTRACTBLASTTHREADCONTROLLALGORITHM initializes the searches and ensures that the requirements for server loading are fulfilled. **Left**: Specialized classes for sending subsequences. **Right**: Specialized classes for sending*ORF* sequences.

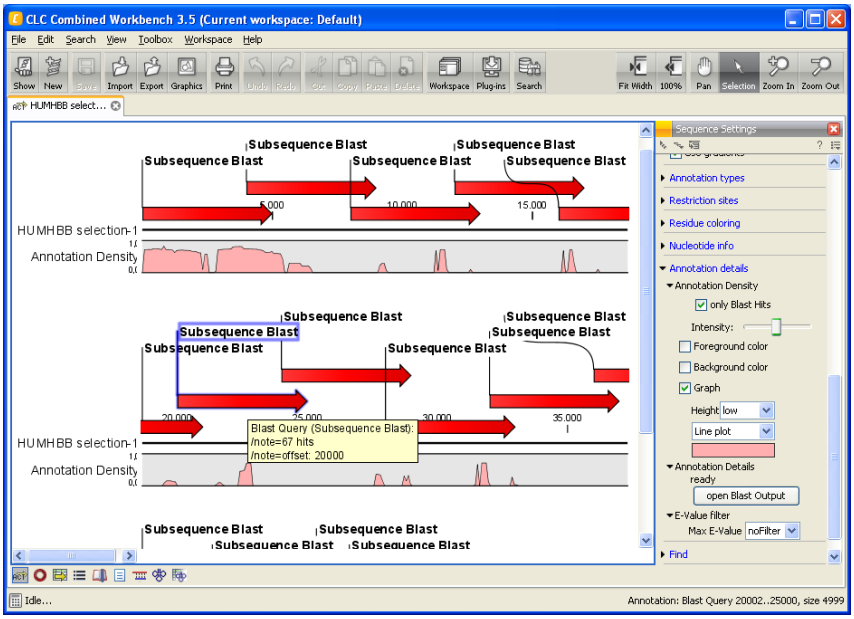

Figure 18: The annotated sequence opened in its default view. Only the subsequences used for the *BLAST* searches are shown as annotations. For a selected subsequence, the corresponding BLASTOUTPUT can be opened using the button on the right side. The graph below the sequences shows the distribution of hits.

6 Evaluation

For the transfer and manual postprocessing of the annotations, the user has full control over the process and the system merely supports him. Therefore, the quality of the result depends highly on the biological knowledge of the user and the time he is willing to spend. However, the splitting of the sequence and the merging of the *BLAST* results is completely automated and can therefore be evaluated.

6.1 Concerning the Time

As shown in Section 2.3.4, the calculation time for a *BLAST* search is linearly dependent on the size of the *query sequence*. Therefore, splitting the sequence in subsequences and performing a *BLAST* search for each of them has no advantage in comparison to a *BLAST* search with the whole sequence. Quite the contrary, the computational costs are even higher as the subsequences are overlapping. However, using subsequences makes it possible to run the searches parallel, and so the actual needed time can be much less as shown in Figure 19. Note that this comparison considers only the *BLAST* search itself and not the time for sending requests or merging of the results.

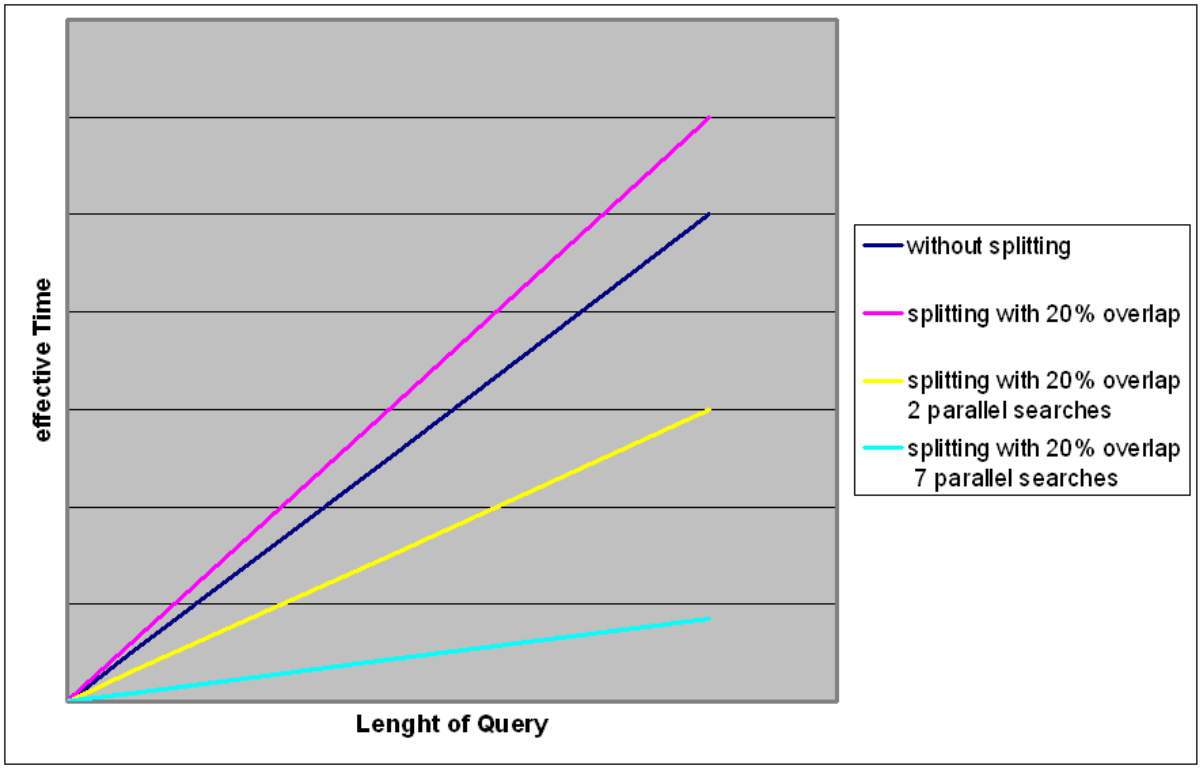

Figure 19: Comparison of theoretical needed time for a *BLAST* search using subsequences and a traditional search without splitting. **Dark Blue:** The whole sequence is used. **Pink:** The sequence is split in subsequences with an overlap of 20%. This is for instance equivalent to a subsequence size of 5000bp and an overlap of 1000bp. All searches are performed one after another. **Yellow:** Subsequneces with 20% overlap and two parallel running searches. **Blue-green:** Subsequneces with 20% overlap and seven parallel running searches.

Comparison for a real query

In order to get an overview of the effective needed time for a real query, a 50*kbp* long region from human chromosome 11 is searched against *RefSeq Protein Database*. To keep the results comparable, all searches are performed on the *NCBI www-blast Server*. However, this server is also used by other clients, and so the spent time depends also on the server load. To reduce the influence of this noise, all searches were repeated 3 times and only the arithmetic mean of the individual needed time is considered. Figure 20 shows the time for the search with the whole sequence as *query sequence* and several searches with different sizes for the subsequences.

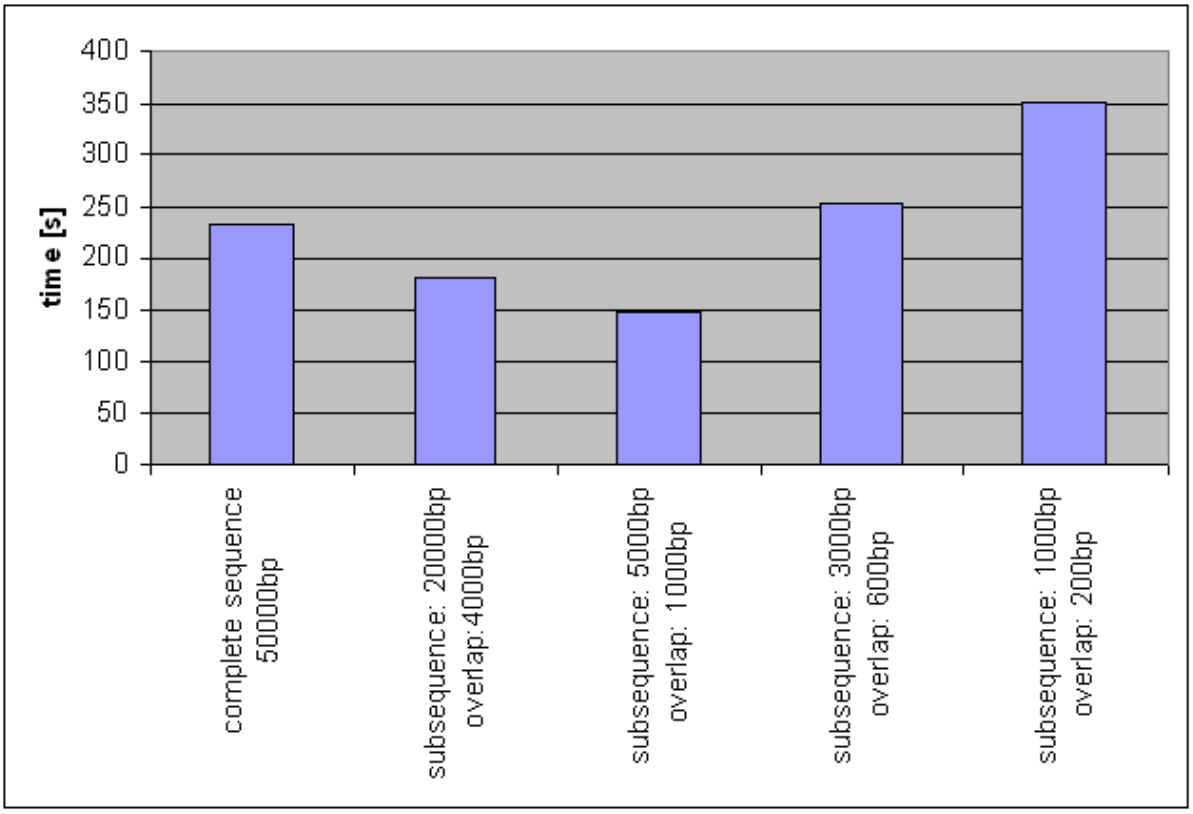

Figure 20: A 50kbp sequence is searched against *RefSeq Protein Database* on the *NCBI www-blast Server*. First, the whole sequence is used for a single search. In all the other cases the sequence is split in subsequences and these are sent to the server. The overlap is always 20% of the subsequence size. To meet the conditions of the *NCBI www-blast Server*, between two sent requests, a time of 3 seconds was guaranteed and never more than 7 parallel requests running.

According to the theoretical consideration in Figure 19 the only relevant parameter for the time for a search is the number of parallel processed subsequences. But Figure 20 indicates that there are further aspects that influence the speed:

- One reason why too short subsequences increase the overall needed time is the required minimum time between two requests. In the searches in Figure 20 this time was set to 3 seconds and the maximum number of running threads was set to 7. If a single search takes less than the product of this two parameter, i.e. 21 seconds, the limiting factor is not

the maximum number of threads but the waiting time. So less than the maximum allowed threads are running.

- As mentioned in Section 2.3.3 the communication between the *CLC bio Workbench* and the *NCBI www-blast Server* is based on *HyperText Transfer Protocol (HTTP)* requests. To avoid timeouts, the server does not wait with its response until the search is completed. It rather sends at once an estimation of the time the search will take. This estimation is usually between 5 and 25 seconds. After this time, a new request has to be sent and the response is either a revised time-estimation or the result of the search. So if the search is completed by the server, it has to wait until the client polls. The client however will not send a new request before it waited the estimated time. Therefore, the client will consider a thread as running even if it is waiting. For a search with a long *query sequence*, the server clearly underestimates the time, so the client will in most cases send several new requests before the search is finished, and will so retrieve an updated estimation. Therefore the unnecessarily waited time is not dependent on the size of the *query sequence* but approximately the same for each request. As a smaller size for subsequences leads to a greater number of requests, it also leads to more waiting time.

- The number of created subsequences depends on the length of each subsequence, the length of the overlap and the length of the complete query as shown in Equation 6. If the length of a subsequence and the length of the complete sequence have the same order of magnitude, only a few subsequences are created. In the this case the edge effects are no more negligible: At the beginning only one thread starts instantly and the next one starts not before the minimum time between two requests is waited. Also at the end not all threads stop at the same time. If the search for the last subsequence is started the number of running threads decreases with every finished thread until the last thread finished. In the extreme case, there are less subsequences in total than available threads. This leads to the circumstance that some threads are not used at all. This is the case for the 20*kbp* subsequences in Figure 20.

$$number\_of\_subsequences = \frac{length\_of\_complete\_sequence}{length\_of\_subsequence - length\_of\_overlap} \tag{6}$$

As the *NCBI www-blast Server* can only handle sequence not considerably longer than 50*kbp*, the comparison was made using a sequence of this size. Though, the used subsequence size is an important factor as there is only a small window with minimum negative influence caused by too short or too long subsequences. This explains the discrepancy between the idealized Figure 19 and the real Figure 20. For longer sequences however, this window is much larger. So several test runs for a 2*Mega base pairs (mbp)* sequence from human chromosome 21 completed within less than an hour. The size for subsequences was set to 10*kbp* with an overlap of 2*kbp*. Not more than seven parallel running requests were allowed and the minimum time between sending two requests was set to 3 seconds.

Time for merging

In the way the merging process is now implemented, it simply deletes duplicate or truncated *HSPs* and corrects some attributes like the region and reading frame for the remaining ones.

For the above mentioned search with a 2*mbp* sequence this took less than two minutes, and is therefore negligible compared to the time for the *BLAST* search itself.

6.2 Concerning the Content

The merged result is not absolutely identical with the result of a single blast search using the whole sequence. This has several reasons:

- As shown in Section 2.3.2 the *e-value* for an *HSP* is linear dependent on the size of the used *query sequence*. As the subsequences are shorter than the whole sequence, the same alignment leads to a lower *e-value*. Therefore, for the same *e-value* threshold more *HSPs* are produced in the split approach. This is only a minor issue, as this can easily be corrected by selecting a more restrictive *e-value* or in a postprocessing step.
- With default settings, *BLAST* returns only 100 hits. So if a region produces lots of very good hits, it can happen that weaker hits in in other regions are not shown as the maximum number of hits is already reached. In the split approach, this counter is for each subsequence reset, so regions in different subsequences are independently considered.
- For consistently ordered hits, *BLAST* applies group statistic. So it is possible for an *HSP* to be classified as significant in the context of other *HSPs* even if it alone would be discarded. However, if this *HSP* is part of another subsequence as the remaining *HSPs* of the group, there is no indication for the existence of this *HSP*. One way to reduce this effect is to use a large overlap, as this makes it less likely for such an *HSP* to be separated from its group.

7 Discussion

7.1 Result of this Project

The result of this project is a software module that extends the *CLC bio Workbench* with the functionality of a semi-automated annotation of megabase-sized DNA sequences by homology search. To annotate an unknown sequence by using this module, several steps have to be performed:

1. Create a new project for the unknown sequence
2. Perform a *BLAST* search against *RefSeq Protein Database*
3. Add the result of the search as annotation to a clone of the unknown sequence
4. Download all protein sequences with a significant hit (hitsequneces)
5. Transfer annotations from the hitsequences to a clone of the unknown sequence
6. Check the annotations manually in the graphical viewer and perform post-processing

For Step 1 a new item in the toolbox menu of the *CLC bio Workbench* was created. Steps 2, 3, 4, 5 are directly initialized in the editor for the project shown in Figure 13. The parameters are set in wizards and no further user action is necessary. It is also possible to repeat step 5 several times with different parameters and compare the different annotations of the created clones. For Step 6, the default SEQUENCEEDITOR of the *CLC bio Workbench* was extended by some tools. For the *BLAST* search, the sequence is split in overlapping subsequences and a own *BLAST* search is performed for each of them. This has 3 advantages:

- The individual searches for the subsequences meet the criteria to run on the *NCBI www-blast Server*.
- As the search is split in smaller independent searches, they can be performed parallel.
- Regions with many very good hits are less likely to suppress weaker hits in other regions.

The basic functionality for the communication with the *NCBI www-blast Server* is already given by the used api. However, the creation of the overlapping subsequences and the management of the parallel requests are part of this project. An important part is to avoid an overload of the server. This is ensured by keeping the number of parallel running searches below a maximum and guarantee a minimum time between two requests. The used *API* provides also basic functionality for the download of sequences from the *NCBI*, which are used with minor modifications.
There are two different possibilities to annotate the sequence. The first one simply adds all information produced by the *BLAST* searches. The second one considers the annotations of the *hit sequences*. These are filtered by various criteria and transferred to the unknown sequence.
For the manual postprocessing step several tools are provided. So it is possible to create a graph that gives a fast overview how the annotations or hits are spread on sequence. Furthermore, the transferred annotations can be filtered by the *e-value* of the corresponding *HSP*. A further feature provides an easy way to access the original results of the search for a single subsequence.

Modified version

The part of the project that manages the parallel requests is designed in a way that it can also be used for other sequences than overlapping subsequence. So a variation was created that takes all *ORFs* of a sequence and performs a *BLAST* search for each of them. This requires that informations about *ORFs* are already added as annotation to the sequence, what can easily be done with an already existing feature of the *CLC bio Workbench*.

7.2 Improvement Opportunities

There are several opportunities how this project can be improved. Some of them are pointed out in the following.

Set up own server

Up to now the *BLAST* search is performed on the *NCBI www-blast Server*. This was fine for the development, and may be also sufficient in cases where the module is only occasionally used. However, for the more extensive use it is recommended that a own server is set up. All the necessary software and databases are provided by the *NCBI*, so this server can provide the same interfaces as the original *NCBI www-blast Server*. As in the *CLC bio Workbench* the URL to the server is not hard coded but defined in the preference settings, also the module created in this project is ready to use such a server.

Recalculation of group statistics

At the moment group statistics for the *HSPs* are not corrected. Rather the size of the overlap is set comparable high to reduce the negative effects. However, the correction of group statistic is theoretical possible, even if it has very high costs. The two main problems are described in following:

- The *HSPs* that are separated from their corresponding group are not necessarily reported as they are classified as non significant due to their high individual *expectation value*. Therefore the *expectation value* threshold for the complete search has to be set less restrict and irrelevant *HSPs* have to be discarded.

- As shown in the appendix, the statistics section of the *XML* output for a *BLAST* search differs from the textual output not only in format but also in content. However, for applying group statistics some of the values that are not part of the *XML* output are essential. So, it would be necessary to parse the textual output, what is obviously not a nice solution.

Ranking of annotations

The decision whether or not an annotation of the *hit sequence* is transferred to the unknown sequence is currently made by a simple filter system. This is based on attributes of the annotation itself like the type, as well as on properties of the corresponding *BLAST* hits like the species or the *e-value* of the *HSPs*. This filtering treats each annotation separately. Therefore, on some regions all annotations may be discarded and on the other hand there a regions with an annotation overkill. A more effective approach would be a ranking system that calculates a specific score for each annotation. So, on each region only the annotation with the highest rank are transfered.

The criteria used for the filtering can be reused in a slightly modified way: The return value is no longer either “transfer” or “discard” but a score. Also further criteria are thinkable for the ranking:

- An annotation is not considered separately but in the context of all potential annotations for the same region on the unknown sequence. If there are two or more equal annotations, they could be merged to one with a higher ranking. On the other hand, contradictory annotations could decrease the ranking of each other. However, this requires a precise definition for “contradictory annotations”.
- Also taxonomic information can be used for the ranking. The actual used filter considers already the species, but for different species the information whether they are of the same family, order, class and so forth is not given. A good resource for such information is the *NCBI* taxonomy database [24].
- For the ranking of an annotation, it can also be relevant if the alignment of the corresponding *HSP* covers a conserved domain. Therefore, a further improvement of the ranking could be made by using resources like the Conserved Domain Database[25].

7.3 Conclusion

With the module developed in this project, the search for protein coding regions of a chromosome-sized *DNA* sequence can be done in a fast, easy and user friendly way. Moreover, possible biological functions of these regions are given. However, the annotations assigned with this module in the actual stage of development can only give a rough overview. For a more exact annotation the used methods have to be improved and combined with further approaches.

Appendices

A References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000.

[2] A. M. Maxam and W. Gilbert. A new method for sequencing dna. *Proc Natl Acad Sci U S A*, 74(2):560–564, Feb 1977.

[3] F. Sanger, S. Nicklen, and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, 74(12):5463–5467, Dec 1977.

[4] Thomas Wiehe Bernhard Haubold. *Introduction to Computational Biology*. Birhäuser, 2006.

[5] Catherine Math, Marie-France Sagot, Thomas Schiex, and Pierre Rouz. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Res*, 30(19):4103–4117, Oct 2002.

[6] J. M. Claverie, O. Poirot, and F. Lopez. The difficulty of identifying genes in anonymous vertebrate sequences. *Comput Chem*, 21(4):203–214, 1997.

[7] J. W. Fickett and C. S. Tung. Assessment of protein coding measures. *Nucleic Acids Res*, 20(24):6441–6450, Dec 1992.

[8] S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, Sep 1997.

[9] Scott McGinnis and Thomas L Madden. Blast: at the core of a powerful and diverse set of sequence analysis tools. *Nucleic Acids Res*, 32(Web Server issue):W20–W25, Jul 2004.

[10] Joseph Bedell Ian Korf, Mark Yandell. *BLAST - An Essential Guide to the Basic Local Alignment Tool*. O*Reilly, 2003.

[11] NCBI. The statistics of sequence similarity scores. http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html. [Online; accessed 29-December-2007].

[12] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, Mar 1981.

[13] E. G. Shpaer, M. Robinson, D. Yee, J. D. Candlin, R. Mines, and T. Hunkapiller. Sensitivity and selectivity in protein similarity searches: a comparison of smith-waterman in hardware to blast and fasta. *Genomics*, 38(2):179–191, Dec 1996.

[14] S. Karlin and S. F. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc Natl Acad Sci U S A*, 90(12):5873–5877, Jun 1993.

[15] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning dna sequences. *J Comput Biol*, 7(1-2):203–214, 2000.

[16] NCBI. Qblast's url api. user's guide. http://www.ncbi.nlm.nih.gov/BLAST/Doc/urlapi.pdf. [Online; accessed 30-December-2007].

[17] NCBI. Blast xml output. ftp://ftp.ncbi.nlm.nih.gov/blast/documents/xml/README.blxml. [Online; accessed 29-December-2007].

[18] W3C. Extensible markup language (xml) 1.1 (second edition). http://www.w3.org/TR/xml11/. [Online; accessed 31-October-2007].

[19] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res*, 35(Database issue):D61–D65, Jan 2007.

[20] *The NCBI handbook Chapter 17, The Reference Sequence (RefSeq) Project.* Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information;2002 Oct. Available from http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Books.

[21] A. Darling, L. Carey, and W. Feng. The design, implementation, and evaluation of mpiblast. *4th International Conference on Linux Clusters: The HPC Revolution 2003*, 2003.

[22] David R Mathog. Parallel blast on split databases. *Bioinformatics*, 19(14):1865–1866, Sep 2003.

[23] J. Zhang and T. L. Madden. Powerblast: a new network blast application for interactive or automated sequence analysis and annotation. *Genome Res*, 7(6):649–656, Jun 1997.

[24] D. L. Wheeler, C. Chappey, A. E. Lash, D. D. Leipe, T. L. Madden, G. D. Schuler, T. A. Tatusova, and B. A. Rapp. Database resources of the national center for biotechnology information. *Nucleic Acids Res*, 28(1):10–14, Jan 2000.

[25] Aron Marchler-Bauer, John B Anderson, Myra K Derbyshire, Carol DeWeese-Scott, Noreen R Gonzales, Marc Gwadz, Luning Hao, Siqian He, David I Hurwitz, John D Jackson, Zhaoxi Ke, Dmitri Krylov, Christopher J Lanczycki, Cynthia A Liebert, Chunlei Liu, Fu Lu, Shennan Lu, Gabriele H Marchler, Mikhail Mullokandov, James S Song, Narmada Thanki, Roxanne A Yamashita, Jodie J Yin, Dachuan Zhang, and Stephen H Bryant. Cdd: a conserved domain database for interactive domain family analysis. *Nucleic Acids Res*, 35(Database issue):D237–D240, Jan 2007.

B Glossary

ab initio gene finding
Methods for identifying genes in a sequence. Mainly based on signals and content of the sequence. 6, 7, 42

actual length
The number of letters of a sequence or a list of sequences. See also *effective length*.. 10, 42

annotation
An additional information about a sequence or a region on a sequence. The term can also refer to the process of adding annotations to a sequence. . 22, 42

Basic Local Alignment Search Tool
a software package used for homology search between biological sequences. 8, 42, 44

bit score
The *raw score* that have been normalized with respect to the scoring system.. 10, 42

BLAST database
a sequence or a list of sequences that is preformated for a *Basic Local Alignment Search Tool* search.. 8, 10–12, 17–19, 21, 22, 42, 43, 46

BLAST query
a sequence or a list of sequences. Each sequence is used as a *query sequence* in an own BLAST search. 13, 17, 42

CLC bio
A company based in Aarhus, Denmark. The scope of business covers the whole field of bioinfrmatics including software, hardware as well as consulting. 15, 26, 27, 42

CLC bio Workbench
A bioinformatics software for sequence analysis . 15, 16, 26–28, 35, 37, 38, 42, 46

comparative gene finding
Methods for identifying genes in a sequence. Mainly based on homology search. 7, 42

effective length
The *effective length* minus a value that considers edge effects.. 10, 21, 42

expectation value
A parameter that describes the number of *HSPs* with a score equal or better than the given one that can be expected by mere chance. 8, 14, 29, 38, 42, 44

heuristic method
A method that uses “rules of thumb” to solve a problem. In most cases it finds a good solution very fast, but it does not guarnatee it.. 8, 42, 44

high scoring segment pair
A *MSP* with an *e-value* beolw a defined threshold. 8, 42–44

hit sequence
A sequence in the *BLAST database* for which at least one *high scoring segment pair* was created. In plain text formated BLAST results, the term SUBJECT is used for this sequence.. 8, 11, 14, 16, 22, 23, 26–28, 31, 37, 38, 42

maximum segment pair
A local alignment of two sequences whose score cannot be increased by extending or trimming. 8, 42, 45

NCBI www-blast Server
Server at the *NCBI* that can used to perform *BLAST*. It is freely accessible via an *HTTP* interface. . 12, 13, 16, 19–21, 26, 29, 34, 35, 37, 38, 42, 47

neighborhood
The neighborhood of a word includes all *words* that achieve at least a defined score when compared by the used *scoring matrix*. 8, 12, 42, 44

Open Reading Frame
subsequence of a DNA sequences that does not contain any stop-codons. 6, 30, 42, 45

Protein Reference Sequence Database
Protein part of the *Reference Sequence Database*. 42, 45

query sequence
the sequence that is searched against a *BLAST database*. 8, 12, 13, 17–19, 21, 22, 26, 27, 33–36, 42, 46

raw score
The score of an alignment calculated as the sum of substitution and gap scores. 10, 42

Reference Sequence Database
A curated non-redundant sequence database of genomes, transcripts and proteins . 14, 22, 42, 43, 45

score
In the context of sequence alginments, the score is a measure for the quality of the alignment. It is the sum of the gap scores and substitution scores calculated using the *scoring matrix*. 42

scoring matrix
TODO. 8, 42, 43

sensitivity
The probability in a binary classification to classify something true as true. 42

Smith-Waterman algorithm
An algorithm for local alignment. As it doesn't use any *heuristic methods*, it guarantes to find the optimal alignment.. 42

Software Developer Kit
A set of programs and documentation that allows the creation of software for a defined system.. 15, 27, 42, 45

specivity
The probability in a binary classification to classify something false as false. 42

sum score
A common score for a group of consistently ordered *HSPs*. 11, 42

word
A chain of W letters, where W is the *word-size*. 8, 42–44

word-hit
An exact match between two words, or to the *neighborhood* of a word. 8, 12, 21, 42

word-size
The length of a *word*. 8, 12, 42, 44

C Acronyms and Abbrevations

API
Application Programming Interface. 15, 29, 31, 37, 42

BLAST
Basic Local Alignment Search Tool. 8–14, 17–23, 25–33, 36–38, 42, 43, 46, 47

DNA
Deoxyribonucleic Acid. 6, 16, 17, 25, 28, 39, 42

e-value
expectation value. 8, 10, 11, 20, 23, 31, 36–38, 42, 43

HSP
high scoring segment pair. 8, 10, 11, 14, 19–21, 23, 30, 31, 35–39, 42, 44, 46

HTTP
HyperText Transfer Protocol. 35, 42, 43

kbp
Kilo base pairs. 19, 34, 35, 42

mbp
Mega base pairs. 35, 36, 42

MSP
maximum segment pair. 8, 10, 42, 43

NCBI
National Center for Biology Information. 11, 12, 14, 31, 37–39, 42, 43

ORF
Open Reading Frame. 6, 7, 26, 32, 38, 42, 47

RefSeq Database
Reference Sequence Database. 14, 42

RefSeq Protein Database
Protein Reference Sequence Database. 13, 15–19, 26, 34, 37, 42, 46, 47

RNA
Ribonucleic Acid. 42

SDK
Software Developer Kit. 15, 16, 26, 42

SNP
Single Nucleotide Polymorphism. 6, 42

XML
eXtensible Markup Language. 13, 21, 38, 42

D List of Figures

E Blast Output - Text vs. XML

Blast can return the result in different formats. If any postprocessing is wanted, XML is usually the best choice as it enables an easy parsing. However, the different blast outputs do not only differ in format, but also in content. This section shows some of the differences between the output of a BLASTx search in plain text format and in XML format. For this reason an example Blast search was performed. A selection of the human beta globin region on chromosome 11 was searched against RefSeq. Listing 1 shows the used parameters.

Listing 1: Used Parameters

```
Program = blastx
Database = refseq_protein
Genetic code = 1
Entrez query = All organisms
Low complexity = Yes
Expect value = 1.0E-10
Word size = 3
Matrix = BLOSUM62
Gap cost (open) = 11
Gap cost (extension) = 1
```

E.1 Representation of a Hit

Listing 2 shows the XML output of a with two HSPs. The text output for the same hit is shown in Listung 2. The fact that both HSPs have the same e-value but a different score indicates the group statistics where applied. In the text output also the number of HSPs used for group statistics is given “Expect(2)”.

Listing 2: Hit in XML

```
<Hit>
   <Hit_num>46</Hit_num>
   <Hit_id>gi|82913926|ref|XP_912634.1|</Hit_id>
   <Hit_def>PREDICTED: similar to Hemoglobin subunit beta (Hemoglobin beta chain) (Beta-globin
       ) [Mus musculus] &gt;gi|149258169|ref|XP_001472850.1| PREDICTED: similar to Hemoglobin
       subunit beta (Hemoglobin beta chain) (Beta-globin) [Mus musculus]</Hit_def>
   <Hit_accession>XP_912634</Hit_accession>
   <Hit_len>146</Hit_len>
   <Hit_hsps>
      <Hsp>
         <Hsp_num>1</Hsp_num>
         <Hsp_bit-score>92.8189</Hsp_bit-score>
         <Hsp_score>229</Hsp_score>
         <Hsp_evalue>8.20814e-23</Hsp_evalue>
         <Hsp_query-from>1081</Hsp_query-from>
         <Hsp_query-to>1308</Hsp_query-to>
         <Hsp_hit-from>31</Hsp_hit-from>
         <Hsp_hit-to>105</Hsp_hit-to>
         <Hsp_query-frame>1</Hsp_query-frame>
         <Hsp_hit-frame>0</Hsp_hit-frame>
         <Hsp_identity>45</Hsp_identity>
         <Hsp_positive>59</Hsp_positive>
         <Hsp_gaps>1</Hsp_gaps>
         <Hsp_align-len>76</Hsp_align-len>
         <Hsp_qseq>
         RLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVL //
         GAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRV
         </Hsp_qseq>
         <Hsp_hseq>
         RILTVYPHTKRYFDHFGDFFCA-ATEGNPKMKALGKKMI //
         ESFSEGLQPLDNLNYTFSSLSELHHDKLHMDPENFKL
         </Hsp_hseq>
         <Hsp_midline>
```

```
            R+L VYP T+R+F+ FGD      A  GNPK+KA GKK++
            +FS+GL  LDNL  TF++LSELH DKLH+DPENF++
            </Hsp_midline>
        </Hsp>
        <Hsp>
            <Hsp_num>2</Hsp_num>
            <Hsp_bit-score>40.817</Hsp_bit-score>
            <Hsp_score>94</Hsp_score>
            <Hsp_evalue>8.20814e-23</Hsp_evalue>
            <Hsp_query-from>861</Hsp_query-from>
            <Hsp_query-to>962</Hsp_query-to>
            <Hsp_hit-from>1</Hsp_hit-from>
            <Hsp_hit-to>34</Hsp_hit-to>
            <Hsp_query-frame>3</Hsp_query-frame>
            <Hsp_hit-frame>0</Hsp_hit-frame>
            <Hsp_identity>18</Hsp_identity>
            <Hsp_positive>26</Hsp_positive>
            <Hsp_gaps>0</Hsp_gaps>
            <Hsp_align-len>34</Hsp_align-len>
            <Hsp_qseq>
                MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLVS
            </Hsp_qseq>
            <Hsp_hseq>
                MVELTAEEKAAITATWTKVKAEELGVESLERILT
                </Hsp_hseq>
            <Hsp_midline>
                MV LT EEK+A+TA W KV  +E+G E+L R+++
            </Hsp_midline>
        </Hsp>
    </Hit_hsps>
</Hit>
```

Listing 3: Hit in plain text

```
>ref|XP_912634.1| PREDICTED: similar to Hemoglobin subunit beta (Hemoglobin beta
chain) (Beta-globin) [Mus musculus]
 ref|XP_001472850.1| PREDICTED: similar to Hemoglobin subunit beta (Hemoglobin beta
chain) (Beta-globin) [Mus musculus]
Length=146

 Score = 92.8 bits (229),  Expect(2) = 8e-23
 Identities = 45/76 (59%), Positives = 59/76 (77%), Gaps = 1/76 (1%)
 Frame = +1

Query  1081  RLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLS  1260
             R+L VYP T+R+F+ FGD      A  GNPK+KA GKK++ +FS+GL  LDNL  TF++LS
Sbjct  31    RILTVYPHTKRYFDHFGDFFCA--ATEGNPKMKALGKKMIESFSEGLQPLDNLNYTFSSLS  89

Query  1261  ELHCDKLHVDPENFRV  1308
             ELH DKLH+DPENF++
Sbjct  90    ELHHDKLHMDPENFKL  105


 Score = 40.8 bits (94),  Expect(2) = 8e-23
 Identities = 18/34 (52%), Positives = 26/34 (76%), Gaps = 0/34 (0%)
 Frame = +3

Query  861  MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLVS  962
            MV LT EEK+A+TA W KV  +E+G E+L R+++
Sbjct  1    MVELTAEEKAAITATWTKVKAEELGVESLERILT  34
```

E.2 Statistics Section

Listing 4 and 5 show the statistics section for this search in two different formats. Some of the data in the plain text can also be found in the XML output. The number of letters and sequences

in the database is similar in both outputs, as well as kappa, lambda, and the entropy. However, all other information shown in the plain text output has either no corresponding element in the XML output, or the value is wrongly replaced by zero.

Listing 4: statistics section in XML

```
<Statistics>
  <Statistics_db-num>3840538</Statistics_db-num>
  <Statistics_db-len>1386750453</Statistics_db-len>
  <Statistics_hsp-len>0</Statistics_hsp-len>
  <Statistics_eff-space>0</Statistics_eff-space>
  <Statistics_kappa>0.041</Statistics_kappa>
  <Statistics_lambda>0.267</Statistics_lambda>
  <Statistics_entropy>0.14</Statistics_entropy>
</Statistics>
```

Listing 5: statistics section in plain text

```
  Database: NCBI Protein Reference Sequences
    Posted date:  Dec 28, 2007  2:21 PM
  Number of letters in database: 1,386,750,453
  Number of sequences in database:  3,840,538
Lambda     K      H
   0.318    0.134    0.401
Gapped
Lambda     K      H
   0.267   0.0410    0.140
Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Number of Sequences: 3840538
Number of Hits to DB: 403928687
Number of extensions: 8691472
Number of successful extensions: 19565
Number of sequences better than 1e-10: 0
Number of HSPs better than 1e-10 without gapping: 0
Number of HSPs gapped: 19446
Number of HSPs successfully gapped: 0
Length of query: 2722
Length of database: 1386750453
Length adjustment: 140
Effective length of query: 2582
Effective length of database: 849075133
Effective search space: 651240627011
Effective search space used: 651240627011
T: 12
A: 40
X1: 16 (7.3 bits)
X2: 38 (14.6 bits)
X3: 64 (24.7 bits)
S1: 41 (20.4 bits)
S2: 177 (72.8 bits)
```