GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

ISSN 1612-6793

Bachelor's Thesis

submitted in partial fulfillment of the
requirements for the course "Applied Computer Science"

My Title

Robin William Hundt

Institute of Computer Science

Bachelor's and Master's Theses
of the Center for Computational Sciences
at the Georg-August-Universität Göttingen

09. May 2020

Georg-August-Universität Göttingen
Institute of Computer Science

Goldschmidtstraße 7
37077 Göttingen
Germany

☎ +49 (551) 39-172000
FAX +49 (551) 39-14403
✉ office@informatik.uni-goettingen.de
www.informatik.uni-goettingen.de

First Supervisor: Prof. Dr. Burkhard Morgenstern
Second Supervisor: Dr. Peter Meinicke

I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, 09. May 2020

Abstract

Motivation

Results

Contents

Chapter 1

Introduction

Chapter 2

Basics

- what is a MSA and for what is it important?
- math def of consistency and msa

2.1 Multiple sequence alignment

As a generalisation of pairwise sequence alignments, multiple sequence alignments are the basis for numerous further analyses such as "inferring phylogenetic relationships, homology search of functional elements, classification of proteins, designing detection markers" [1, pg. 3]. In contrast to the pairwise alignment problem, aligning an arbitrary number of sequences is a NP-complete problem, when formulated as the maximisation (or minimisation) of an objective function [1, pg. 172].
The classical formulation of the problem is based upon a model of evolution where single residues get inserted, deleted or substituted. Since an insertion in one sequence is indistinguishable of a deletion in another one, these two operations are commonly viewed as one and referred to as an *indel*. The goal is then to insert gaps, representing an indel and denoted as '-' into the sequences, such that they have the same length and a given score function is maximal for the produced Alignment. These aligned sequences are usually displayed as a table of residues and gap characters as seen in table 2.1 [1].

seq1	L	L	I	R	N	L	I	Q	V	-	V	K	S	V	-	-	-	-
seq2	L	L	I	R	K	L	I	D	V	-	V	R	T	V	-	-	-	-
seq3	L	L	I	R	Q	L	I	D	V	-	I	K	T	V	-	-	-	-
seq4	L	L	I	-	-	-	-	Q	M	A	D	Q	Y	L	P	E	T	L

Table 2.1: Example of multiple sequence alignment. The gap symbol '-' represents an insertion or deletion (often combined as *indel*).

2.2 Mathematical notion of Consistency and Alignments

A more formal definition of the term *Alignment* is based upon the work by Morgenstern et al. [2] and Abdeddaïm [3]. The following definitions constitute a condensed formalization of the one provided in the preliminary work to this thesis [4].

Let S_i be a Sequence over an Alphabet, e.g. DNA, Amino Acids, etc., and $S := \{S_1, ..., S_n\}$ a set of Sequences.

Definition 2.2.1 (Site)
A site $x = [i, p]$ represents the p-th position in the i-th sequence and X is the set of all sites for S.
The function

$$\begin{aligned} seq :& X \to \mathbb{N} \\ & x \mapsto i \end{aligned}$$

maps a site to its corresponding sequence. Whereas the function

$$\begin{aligned} pos :& X \to \mathbb{N} \\ & x \mapsto p \end{aligned}$$

maps a site to its position.

Definition 2.2.2 (Ordering of Sites)
For $x = [i, p], x' = [i', p'] \in X$ we define $x \preceq y$ if and only if $i = i'$ and $p \leq p'$.
The relation $\preceq$ is a partial ordering on X.

Lemma 2.2.1 (Extension of binary relation to quasi order relation)
Let A be a reflexive binary relation on some set X and R any binary relation on X.
The transitive closure $\preceq_R := (A \cup R)^t$ is a quasi order relation on X.

As per lemma 2.2.1 we extend $\preceq$ to the quasi partial order $\preceq_R$ by taking the transitive closure of the union of $\preceq$ and R. Consequently $u \preceq v$, vRw, $w \preceq x$, xRy and $y \preceq z$ from which follows that $y \preceq_R z$.

Definition 2.2.3 (Consistency)
Let R be a binary relation on a set of sites X. R is consistent if for $x, y \in X$ where $seq(x) = seq(y)$

$$x \preceq_R y \implies x \preceq y$$

holds. Additionally a set $\{R_1, ..., R_n\}$ of binary relations on X is consistent if $\cup_i R_i$ is consistent.

Definition 2.2.4 (Alignment)
An alignment (or partial alignment) A is a consistent equivalence relation on the set of sites X for a set of sequences S.

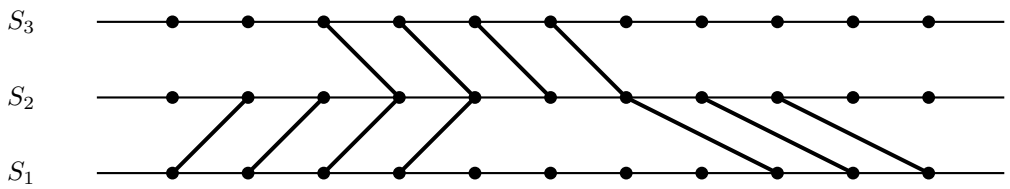

Figure 2.1: Example of a consistent equivalence relation A on the set of sites X for $S = \{S_1, S_2, S_3\}$. Connections between nodes x, y represent represent a relation xAy.

Chapter 3

Prior Work

Spam-Align align builds upon several other algorithms and methods, namely GABIOS-LIB by Abdeddaïm [5], Dialign by Morgenstern et al. [2] which later incorporated GABIOS-LIB [3] as well as Spaced Word Matches proposed by Leimeister et al. in *Fast alignment-free sequence comparison using spaced-word frequencies* [6].

3.1 GABIOS-LIB

GABIOS-LIB is a library written by Saïd Abdeddaïm implementing the *EdgeAddition* algorithm for the incremental computation of the transitive closure of an alignment graph [5].

- written to be used in greedy alignment algorithms, used in Dialign 2.2
- able to answer the question of whether two sites are alignable in $o(1)$
- allows efficient incremental updates of closure when new sites are added to alignment
- $O(k^2n + n^2)$ time for k sequences with a total length of n ($n = |X|$)
- $O(kn)$ space

3.2 Dialign

Dialign is a multiple sequence algorithm first proposed by Morgenstern et al. in 1996 [2] with several alterations and reimplementations over the years [3,7–10]. In the following, *Dialign* will refer to the version 2.2 [10] of the software.

- is an iterative and greedy algorithm

- first computes pairwise alignments consisting of a set of non overlapping segment pairs (referred to as diagonals) with a maximal sum of scores for every combination of sequences
- set of diagonals is scored by their weight and greedily incorporated into alignment while using GABIOS-LIB to efficiently ensure consistent of alignments
- diagonals with score below threshold (default 0) are discarded
- process of finding diagonals and adding them to alignment is repeated for unaligned parts of sequences until everything is aligned or no diagonals of positive score can be found
- dialign sepnds majority of time on constructing paiwise alignments in order to find diagonals -> src is own lousy profile, maybe show profile data for big alignment in bb in appendix?

3.3 Spaced Word Matches

- based on spaced seeds by [11]
- pattern of care and don't care positions is used to find imprecise? matches between sequences

S_1:	a	b	l	l	h	i	a	f	c	b
S_2:	c	b	l	i	g	i	k	f	i	t
P:		1	1	0	0	0	0	1		

Table 3.1: Example of a Spaced Word Match

Definition 3.3.1 (Pattern as defined in [4])
A Pattern is a sequence over the Alphabet $\Sigma = \{0, 1\}$*, where* 1 *corresponds to a "Match position" and* 0 *to a "Don't Care" position. A patterns weight* k *is defined as the number of "Match positions" it contains.*

3.3.1 Multi dimensional matches

S_1:	a	b	l	l	h	i	a	f	c	b
S_2:	c	b	l	i	g	i	k	f	i	t
S_3:	k	b	l	q	k	i	a	f	i	l
P:		1	1	0	0	0	0	1		

Table 3.2: Example of a multi dimensional Spaced Word Match

Chapter 4

Algorithm

This thesis provides an implementation and improvement of the alignment algorithm proposed in [4].

Algorithm 1: align(S, P)

Data: Sequences $S = S_1, ..., S_N$
Data: Pattern set $P = P_1, ..., P_M$
Result: Partial Alignment A
partial_alignment $\leftarrow \{\}$
micro_alignments $\leftarrow$ `find_spaced_word_matches(`S, P`)`
micro_alignments.`sort_by_score_descending()`
foreach *ma in* micro_alignments **do**
    if `is_inconsistent(`*ma*, partial_alignment`)` **then**
        continue
    foreach *site_pair in ma* **do**
        if partial_alignment.`not_aligned(`*site_pair*`)` **then**
            partial_alignment.`add_site_pair(`*site_pair*`)`
return partial_alignment

Observation 3 of [3,5] gave rise to following algorithm which can be improved

TODO Question: Why are 2 and 3 equivalent?

As pointed out in [3,4] partitioning the aligned sites into equivalence classes respective to the relation of whether two sites aligned allows for a more compact storage and efficient update of the transitivity frontiers due to them coinciding for every site in an eq class.

Algorithm 2: add_site_pair(x, y) as proposed in [4]

Data: Consistent site pair $a, b \in X$ to align
Data: Partial Alignment A
Result: Partial Alignment $A' = A \cup \{(a, b)\}$
// Clone the old pred and succ values
$pred \leftarrow pred_A$
$succ \leftarrow succ_A$
// Update the successor frontier
foreach $x \in X$ **do**
  for $i \leftarrow 1$ **to** N **do**
    if $x \preceq_A a$ **then**
      $succ_A[x, i] \leftarrow \texttt{min}\,(succ[x, i], succ[b, i])$
    else if $x \preceq_{A_i} b$ **then**
      $succ_A[x, i] \leftarrow \texttt{min}\,(succ[x, i], succ[a, i])$
    else
      $succ_A[x, i] \leftarrow succ[x, i]$
// Update the predeccessor frontier
foreach $x \in X$ **do**
  for $i \leftarrow 1$ **to** N **do**
    if $x \succeq_A a$ **then**
      $pred_A[x, i] \leftarrow \texttt{max}\,(pred[x, i], pred[b, i])$
    else if $x \succeq_{A_i} b$ **then**
      $pred_A[x, i] \leftarrow \texttt{max}\,(pred[x, i], pred[a, i])$
    else
      $pred_A[x, i] \leftarrow pred[x, i]$

While the algorithm 2 for maintaining a transitive closure when adding a pair of sites (a, b) into a partial alignment works, it is far from optimal.

Figure 4.2 displays a partial alignment between two sequences. The continuous line represents two sites already aligned while the dotted line connects the site pair that is to be added. Updating the predecessor frontier for sequence S_1 would result in the following updates to $pred_A[x, i]$:

It is evident that once the position of x is greater or equal than that of an already aligned position, the alignment of a and b has no effect on the predecessor frontiers $pred_A[x, 2]$ for sites x with a position that is greater or equal than 3.
This property allows an alternative solution

- describe version of algorithm utilizing eq classes and property described in 4.2 -> should this include data structures and memory evincemanagement?
- maybe first dp algorithm without unnecessary cmps (like in Fi. 4 of [5]) and the formulate with eq classes (which wasn't done in paper)

Algorithm 3: add_site_pair(x, y) as proposed by Abdeddaïm [5]

Data: Consistent site pair $a, b \in X$ to align
Data: Partial Alignment A
Result: Partial Alignment $A' = A \cup \{(a, b)\}$
`// Update the successor frontier`
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** N **do**
    for $p \leftarrow pred_A[a][i]$ **to** 1 **do**
      $u \leftarrow Site[p, i]$
      if *not* $succ_A[b][j] < succ_A[u][j]$ **then**
        break
      $succ_A[u][j] \leftarrow succ_A[b][j]$
`// Update the predecessor frontier`
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** N **do**
    for $p \leftarrow succ_A[b][i]$ **to** $len(S_i)$ **do**
      $u \leftarrow Site[p, i]$
      if *not* $pred_A[a][j] > pred_A[u][j]$ **then**
        break
      $pred_A[u][j] \leftarrow pred_A[a][j]$

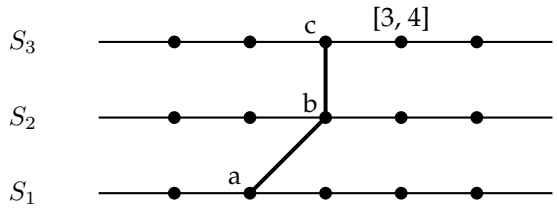

Figure 4.1: TODO

- highlight how this differs from most alignment algorithms in that it is completely greedy

Algorithm 4: revised add_site_pair(x, y) based on GABIOS-LIB implementation in Dialign2.2 [3]

Data: Consistent site pair $a, b \in X$ to align
Data: Index nn of alignment set that was merged from alignment sets of a and b
Data: Successor frontier $succ$
Data: Predecessor frontier $pred$
Data: $alig\_set$ matrix, mapping an alignment set and sequence to a position
Data: $pred\_alig\_set\_pos$ mapping a sequence and position to the index of the next predecessor alignment set of that site
Result: Updated tranisitivity frontiers $succ$ and $pred$

```
// Update the successor frontier
frontier_ops ← [ ]
for i ← 1 to N do
    if left_a[i] == left_b[i] then
        continue
    for j ← 1 to N do
        k ← pred_A[nn, i]
        if k > 0 and k == alig_set[nn, i] then
            k ← pred_alig_set_pos[i, k]
        while k > 0 do
            n ← alig_set_nbr[i, k]
            if succ[n, j] > succ[nn, j] then
                frontier_ops.push ([n, j, succ[nn, j]])
                k ← pred_alig_set_pos[i, k]
            else
                break
foreach [n, j, new_front] ∈ frontier_ops do
    succ[n, j] ← new_front

// Update the predecessor frontier
// ommited for brevity
```

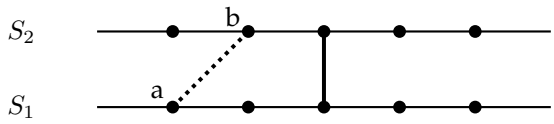

Figure 4.2: TODO

x	$pred_A[x, 2]$ before update	$pred_A[x, 2]$ after update
(1, 1)	0	2
(1, 2)	0	2
(1, 3)	3	3
(1, 4)	3	3
(1, 5)	3	3

Chapter 5

Implementation

The implementation of the algorithm is done in the Rust programming language [1] and contained in the git sub module `spam-align` of the alignment evaluation folder.

The core part of the algorithm as described in 4 is designed to iteratively maintain a partial alignment, as per definition 2.2.4, that allows the fast insertion of newly aligned sites as well as checking if a given pair of sites is consistent with the given alignment.
An initial implementation is part of the *Dialign2.2* program

- explain

```
pub struct Closure {
 sequences: Sequences,
 alig_set: Matrix<usize>,

 nbr_alig_sets: usize,
 old_nbr_alig_sets: usize,

 pred_frontier: Matrix<usize>,
 succ_frontier: Matrix<usize>,

 pred_frontier_ops: Vec<FrontierOp>,
 succ_frontier_ops: Vec<FrontierOp>,
}

struct Sequences {
  lengths: Vec<usize>,
  alig_set_nbr: Matrix<usize>,
  pred_alig_set_pos: Matrix<usize>,
  succ_alig_set_pos: Matrix<usize>,
}
```

Listing 1: Data types responsible for storing partial alignment information.

Listing 1 provides the definition of the core data types responsible for tracking the status of an alignment that is constructed by iteratively aligning sites. A `struct` in Rust functions as simple record of heterogeneous data similar to those in the *C* programming language. Members of a

[1] rust-lang.org/

struct are defined as <name of member field>: <type of field>.
alig_set: **Matrix**<**usize**> is a 2-dimensional matrix implementation containing **usize** elements, which are usigned integers with a size equal to the target architecture pointer size (meaning 64 bits on a 64 bit target).

Differences to Gabios-Lib

- matrices are contiguous memory instead of pointer of pointers -> reduces indirection and improves cache locality
- no unnecessary left and right buffers which need to be written each iteration
- frontier ops instead of pos matrix allows applying frontier changes in $O(\#changes)$ instead of $O(\#seq^2 * \text{part in while loop, don't know upper bound})$
- sequences is struct of matrices instead of Vec of struct of vecs -> less indirection

Chapter 6

Evaluation

6.1 BAliBASE 3 alignment benchmark dataset

The third version of the BAliBASE benchmark protein alignment database has been released in 2005 and is widely employed for the comparison of multiple alignment programs [1, 12]. It is constructed in a semi automatic process as shown in fig. 6.1 and suitable to evaluate global and local alignment programs. The database is split into 5 reference sets with different characteristics representing distinctive multiple alignment problems.
It is divided into:

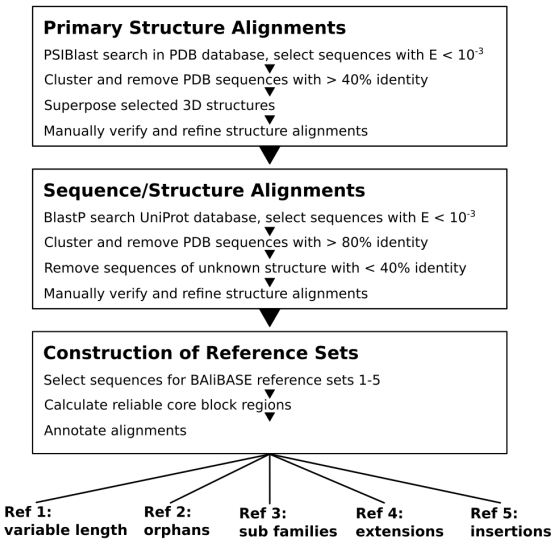

Figure 6.1: Flow chart showing the semi automatic process used to establish the reference sets TODO cite self

- reference set 1 subset V1, for which any two sequences share <20% identity and no internal insertions over 35 residues long
- reference set 1 subset V2, consisting of families with at least four equidistant sequences for which any two sequences share 20-40% identity and no large insertions
- reference set 2, for which all sequences share >40% identity and at least one 3D structure is known. Additionally an "Orphan" sequence with <20% identity is chosen per family
- for reference set 3, all sequences in the same subfamily have >40% identity, whereas sequences from

different subfamilies share <20% identity

- for reference sets 4 and 5, every sequence shares at least 20% with one other sequence, including sequences with large N/C-terminal extensions (ref 4) or internal insertions (ref 5)

6.1.1 Core blocks

Evaluating and comparing alignment programs is a difficult problem due to the uncertainty of supposedly "real" alignments of actual sequences. The BAliBASE database marks alignment columns which can be reliably aligned as so called "core blocks". These core blocks are calculated and manually verified, making up 19% of the full length sequences which are used in the evaluation of *spam-align* [12].

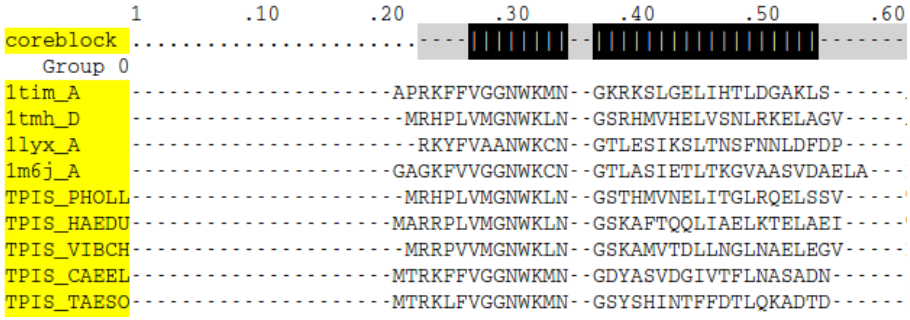

Figure 6.2: BAliBASE web interface. Black columns indicate core blocks. lbgi.fr

6.1.2 Quality of Alignments

Quality measures for protein alignment benchmarks
https://academic.oup.com/nar/article/38/7/2145/3100529

6.2 Sum-of-pairs and column score

Comparing the alignment output of different methods can be done by computing the sum-of-pairs and column scores.
Given a test alignment A_t and a reference alignment A_r with M sequences and N_t, N_r columns respectively, the sum-of-pairs and column score is defined according to Thompson et al. [13].

Definition 6.2.1 (Sum-of-pairs score)
The sum of pairs score is the ratio of correctly aligned individual residues. Formally it is defined as:

$$p_{ijk} = \begin{cases} 1 \textit{ if residues } A_{t_{ij}} \textit{ and } A_{t_{ik}} \textbf{\textit{ are }} \textit{aligned in } A_r \\ 0 \textit{ otherwise} \end{cases}$$

$$S_i = \sum_{j=1}^{M} \sum_{k=i+1}^{M} p_{ijk}$$

$$SPS = \frac{\sum_{i=1}^{N_t} S_i}{\sum_{i=1}^{N_r} S_{r_i}}$$

with S_{r_i} being the number of correctly aligned residues in the reference.

Definition 6.2.2 (Column score)
The column score is the ratio of correctly aligned columns.

$$C_i = \begin{cases} 1 \textit{ if all the residues in the i-th column are aligned correctly} \\ 0 \textit{ otherwise} \end{cases}$$

$$CS = \frac{\sum_{i=1}^{N_t} C_i}{N_r}$$

Note that $C_i = 1$ only if all the residues in the i-th column are aligned correctly and no residue belonging to this column is part of another one. For this reason, the numerator is smaller or equal to the denominator.
The definition of the column score is slightly different than that provided by the authors of BAliBASE [13] but resembles the actual implementation in the included BaliScore tool and its reimplementation provided with this thesis.
These scores are only calculated for the core blocks of the BAliBASE alignments, meaning that for the following evaluation A_t is an alignment over the full sequences, while A_r contains only the aligned residues inside the core blocks.

6.3 Evaluated programs

6.3.1 Mafft

Additionally to *Dialign2.2* and *spam-align* the widely used multiple alignment program *MAFFT* (version 7) is evaluated. It employs a progressive alignment strategy

- progressive alignment
- guide tree from all pairwise alignments

MAFFT is evaluated for two different alignment strategies, from now on referred to as *fast* and *accurate*

6.3.2 Dialign

6.3.3 Spam-Align

6.4 Results

Chapter 7

Conclusion

7.1 Further work

7.1.1 Parallelisation Opportunities

Bibliography

[1] D. J. Russell, *Multiple Sequence Alignment Methods -*, softcover reprint of the original 1st ed. 2014 ed. Humana Press, 2016.

[2] B. Morgenstern, A. Dress, and T. Werner, "Multiple dna and protein sequence alignment based on segment-to-segment comparison," *Proceedings of the National Academy of Sciences*, vol. 93, no. 22, pp. 12 098–12 103, 1996.

[3] S. Abdeddaïm and B. Morgenstern, "Speeding up the dialign multiple alignment program by using the 'greedy alignment of biological sequences library'(gabios-lib)," in *International Conference on Biology, Informatics, and Mathematics*. Springer, 2000, pp. 1–11.

[4] R. W. Hundt, "Praktikumsbericht," *Vertiefte anwendungsorientierte Systementwicklung im forschungsbezogenen Praktikum, Georg-August-Universität Göttingen*, 2020.

[5] S. Abdeddaïm, "On incremental computation of transitive closure and greedy alignment," in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 1997, pp. 167–179.

[6] C.-A. Leimeister, M. Boden, S. Horwege, S. Lindner, and B. Morgenstern, "Fast alignment-free sequence comparison using spaced-word frequencies," *Bioinformatics*, vol. 30, no. 14, pp. 1991–1999, 2014.

[7] B. Morgenstern, "Dialign 2: improvement of the segment-to-segment approach to multiple sequence alignment." *Bioinformatics (Oxford, England)*, vol. 15, no. 3, pp. 211–218, 1999.

[8] A. R. Subramanian, J. Weyer-Menkhoff, M. Kaufmann, and B. Morgenstern, "Dialign-t: An improved algorithm for segment-based multiple sequence alignment," *BMC Bioinformatics*, vol. 6, p. 66, 2005.

[9] A. R. Subramanian, M. Kaufmann, and B. Morgenstern, "Dialign-tx: greedy and progressive approaches for the segment-based multiple sequence alignment," *Algorithms for Molecular Biology*, vol. 3, p. 6, 2008.

[10] B. Morgenstern, "Dialign: multiple dna and protein sequence alignment at bibiserv," *Nucleic acids research*, vol. 32, no. suppl_2, pp. W33–W36, 2004.

[11] B. Ma, J. Tromp, and M. Li, "Patternhunter: faster and more sensitive homology search," *Bioinformatics*, vol. 18, no. 3, pp. 440–445, 2002.

[12] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, "Balibase 3.0: latest developments of the multiple sequence alignment benchmark," *Proteins: Structure, Function, and Bioinformatics*, vol. 61, no. 1, pp. 127–136, 2005.

[13] J. D. Thompson, F. Plewniak, and O. Poch, "A comprehensive comparison of multiple sequence alignment programs," *Nucleic acids research*, vol. 27, no. 13, pp. 2682–2690, 1999.