BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FULLY AUTOMATED METHOD FOR THE DESIGN OF ANCESTRAL PROTEINS

NÁSTROJ PRO AUTOMATIZOVANÝ NÁVRH ANCESTRÁLNÍCH PROTEINŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. MARTIN ŠTĚPÁNEK**
AUTOR PRÁCE

SUPERVISOR **Ing. MILOŠ MUSIL, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Information Systems (DIFS) Academic year 2021/2022

Master's Thesis Specification

24649

Student:	**Štěpánek Martin, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Bioinformatics and Biocomputing
Title:	**Fully Automated Method for the Design of Ancestral Proteins**
Category:	Biocomputing

Assignment:

1. Study the problem of protein engineering and protein stability.
2. Study the evolution-based methods for the design of stable proteins, namely ancestral sequence reconstruction.
3. Study the methods and tools that are necessary for the calculation of the ancestral sequences, the most important methods for the search of homolog sequences, and the construction of the multiple-sequence alignment and phylogenetic tree.
4. Study the ancestral sequence reconstruction technique used by FireProt-ASR.
5. Extend the existing approach with novel techniques such as latent-space search and the successor analysis.
6. Implement your solution together with the interactive user interface.
7. Evaluate the robustness of the method and its results.

Recommended literature:

- Gromiha, MM., Protein Bioinformatics: from sequence to function, ISNB: 978-81-312-2297-3.
- Mazurenko, S., 2020: Predicting Protein Stability and Solubility Changes Upon Mutations: Data Perspective. ChemCatChem 12: 1-10.
- Musil, M., Khan, TK., Beier, A., Stourac, J., Konegger, H., Damborsky, J., Bednar, D., 2020: FireProt-ASR: A Web Sever for Fully Atomated Ancestral Sequence Reconstruction, Briefings in Bioinformatics, 22, 4: bbaa337.
- Musil, M., Konegger, H., Hon, J., Bednar, D., Damborsky, J., 2019: Computational Design of Stable and Soluble Biocatalysts. ACS Catalysis 9: 10331054.

Requirements for the semestral defence:

- First four items of the assignment.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Musil Miloš, Ing., Ph.D.**
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	November 1, 2021
Submission deadline:	May 18, 2022
Approval date:	October 22, 2021

Abstract

Protein stability is a crucial feature for the industrial applications of proteins. This thesis focuses on the enhancements of FireProt^{ASR}, an automated tool for the design of stable proteins via ancestral sequence reconstruction. A novel technique of the successor prediction was developed and implemented into FireProt^{ASR}. Successors represent the evolutionary future of protein sequences and are supposed to have higher activity than current day proteins. They are also expected to have higher ability to target specific molecules. Furthermore, a new web user interface of FireProt^{ASR} was developed. It provides a fast and intuitive way to control the tool.

Abstrakt

Stabilita proteinů je zásadní vlastností ovlivňující možnosti průmyslového použití proteinů. Tato práce se zaměřuje na vylepšení automatizovaného nástroje FireProt^{ASR}, který využívá rekonstrukci ancestrálních sekvencí pro návrh proteinů se zvýšenou stabilitou. Do tohoto nástroje byla přidána nově navržená metoda rekonstrukce successorů. Successory představují sekvence proteinů, ke kterým bude v budoucnosti směřovat evoluce. Podle současných poznatků by měly mít vyšší aktivitu než současné proteiny, společně s vyšší schopností vázat se specificky na určité molekuly. Dále bylo vytvořeno nové webové uživatelské rozhraní, které poskytuje rychlé a intuitivní ovládání nástroje FireProt^{ASR}.

Keywords

protein engineering, protein stability, stability prediction, ancestral sequence reconstruction, successors

Klíčová slova

proteinové inženýrství, stabilita proteinů, predikce stability, rekonstrukce ancestrálních sekvencí, successory

Reference

ŠTĚPÁNEK, Martin. *Fully Automated Method for the Design of Ancestral Proteins.* Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Miloš Musil, Ph.D.

Rozšířený abstrakt

Studium proteinů je jedním z nejdůležitějších úkolů bioinformatiky, neboť proteiny patří mezi základní stavební kameny veškerého života. Hrají také zásadní roli při vývoji a výrobě léčiv. Pro praktické využití proteinů je důležité, aby se s nimi dalo pracovat v podmínkách, které se vyskytují při jejich průmyslové výrobě či aplikaci. Takové podmínky často neodpovídají podmínkám, ve kterých se proteiny vyskytují přirozeně a pro něž jsou evolučně přizpůsobeny.

Je zde proto poptávka po vývoji nových proteinů, které mají stejnou funkci jako existující proteiny, ale mají navíc vyšší stabilitu a jsou tak například použitelné i při zvýšené teplotě. Tyto nové proteiny jsou získávány zaváděním mutací do původních proteinů. Nejspolehlivějším způsobem ověření, jaký vliv má daná mutace na stabilitu proteinu, je exaktní změření stability v rámci laboratorního experimentu. Tato metoda je nicméně nákladná jak finančně, tak časově. Proto vznikají výpočetní nástroje, které predikují vhodné mutace vedoucí ke zvýšení stability proteinů.

Jedním z takových nástrojů je i FireProt^{ASR} vyvíjený v Loschmidtových laboratořích v Brně. Tento nástroj využívá metodu rekonstrukce ancestrálních sekvencí, jejímž cílem je navrhnout sekvence proteinů, které se na Zemi vyskytovaly v minulosti a jsou evolučními předky současných proteinů. Bylo ukázáno, že tyto ancestrální proteiny vykazují obvykle vyšší stabilitu, pravděpodobně proto, že se vyvíjely v drsnějších podmínkách, které v té době na Zemi panovaly. Ancestrální rekonstrukce má několik kroků, které tradičně vyžadují velké množství manuální práce a expertních znalostí. FireProt^{ASR} však nabízí plnou automatizaci celého procesu.

Vstupem nástroje FireProt^{ASR} je jedna sekvence proteinu. K ní je v prvním kroku nalezena množina evolučně příbuzných sekvencí, tzv. homologů. Vyhledávání homologů je prováděno s ohledem na zachování biologické relevance, tak aby nebyla změněna funkce daného proteinu. Z množiny homologů je posléze sestaveno vícenásobné zarovnání, které je použito pro konstrukci fylogenetického stromu. Z něj jsou nakonec získány ancestrální sekvence.

Tato práce se zabývá rozšířením nástroje FireProt^{ASR} o nové techniky a implementací nového uživatelského rozhraní pro tento nástroj. Původně byly navrženy dvě nové metody – predikce tzv. successorů, a použití latentních prostorů pro výpočet ancestrálních sekvencí. Dříve, než byla metoda latentních prostorů přidána do nástroje FireProt^{ASR}, však byla prozkoumána v rámci jiných nástrojů vyvíjených v Loschmidtových laboratořích a bylo zjištěno, že tato metoda nepřináší kýžené výsledky. Proto byla v rámci této práce implementována pouze predikce successorů.

Successory představují opak ancestrálních sekvencí z hlediska směru evoluce. Jsou to tedy sekvence proteinů, ke kterým bude evoluce směřovat v budoucnosti. Oproti rekonstrukci ancestrálních sekvencí je predikce successorů velmi málo prozkoumaná metoda. Poznatky o ancestrálních sekvencích však naznačují, že successory by měly vykazovat vyšší aktivitu a specificitu a nižší stabilitu než současné proteiny. Zmiňovaná nižší stabilita je jistě nevýhodou, ale v některých aplikacích může být výhoda vyšší aktivity důležitější.

Základem predikce successorů, implementované v této práci, je výpočet trendu, neboli směru, kterým se evoluce ubírala od ancestorů po současné proteiny. Tento směr je pak použit pro extrapolaci budoucího vývoje. Důležitým faktorem je vhodná reprezentace aminokyselin. Nakonec byla zvolena metoda, kdy je každá z dvaceti aminokyselin reprezentována pomocí fyzikálně-chemických vlastností z databáze AAindex.

Dalším výsledkem této práce je vytvoření nového backendu nástroje FireProt^{ASR}. Ten představuje prostřední článek, který propojuje výpočetní jádro nástroje s uživatelským

rozhraním. Backend byl implementován v jazyce Java a má podobu HTTP serveru, který zpracovává požadavky od frontendu a zajišťuje tak spouštění výpočetních úloh či stahování výsledných souborů z výpočetního jádra.

Na backend navazuje další výsledek této práce, nový frontend implementující webové uživatelské rozhraní. V původní verzi nástroje FireProt$^{\mathrm{ASR}}$ bylo uživatelské rozhraní implementováno v jazyce Java s využitím frameworku GWT. Toto řešení však skýtalo mnohé nedostatky, ať už ve výkonnosti, či udržovatelnosti kódu. Proto bylo v této práci vytvořeno úplně nové rozhraní v jazyce TypeScript s knihovnou React.

Bylo zjištěno, že predikce successorů přináší výsledky, které jsou konzistentní s výsledky ancestrální rekonstrukce v nástroji FireProt$^{\mathrm{ASR}}$. Přitom rekonstruované ancestrální sekvence byly už dříve podrobeny laboratorním experimentům a bylo prokázáno, že vykazují zvýšenou odolnost vůči teplotě. Podobné experimenty však bohužel nebyly provedeny v případě successorů, neboť laboratorní testy jsou nad rámec této práce. Bylo však ukázáno, že generované sekvence successorů jsou smysluplné a odpovídají zadaným požadavkům.

Nové uživatelské rozhraní je plně funkční a v budoucnu nahradí současné rozhraní v produkční verzi nástroje FireProt$^{\mathrm{ASR}}$. Pro uživatele představuje zrychlení i zjednodušení ovládání celého nástroje.

Fully Automated Method for the Design of Ancestral Proteins

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Miloš Musil, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Martin Štěpánek
May 16, 2022

Acknowledgements

I would like to give my thanks to my supervisor Ing. Miloš Musil, Ph.D for his guidance and suggestions, as well as Josef Kadleček for the valuable discussions concerning this thesis and the meaning of life.

Contents

Chapter 1

Introduction

Proteins are one of the most important molecules in living organisms. They occur in all cells and have many different functions. They are also important for the pharmaceutical industry as they are the most common drug targets. In recent years, many new technologies have helped to deepen our knowledge of proteins in an unprecedented manner. The next-generation sequencing of DNA is just one example of such technologies. The amount of data in databases of biological information experienced exponential growth. Nevertheless, the search space of many important tasks is too big for laboratory experiments. Therefore, computers and effective bioinformatics algorithms must be applied in order to solve some of these tasks.

The stability of proteins defines the ability of proteins to function even in harsh conditions. This feature is tremendously important for the drug design. For example, proteins with a higher stability can withstand the conditions in the industrial environment, such as different pH or a presence of denaturing agents. Therefore, a huge effort is made in order to design proteins with higher stability by introducing suitable mutations into existing proteins.

One of the most promising computational techniques for the design of stable proteins is a phylogenetic analysis, which usually works with a set of homologous sequences and tries to suggest mutations leading to a higher stability while preserving the function of the protein. One of the main methods of the phylogenetic analysis is ancestral sequence reconstruction (ASR). This method attempts to reconstruct extinct protein sequences, which are the evolutionary ancestors of the nowadays proteins.

Ancestral sequence reconstruction is a complex method which requires a lot of biological knowledge and uses a plenty of bioinformatic algorithms. All the relevant concerns are described in the theoretical part in chapters 2 – 6. Chapter 2 introduces the relevant biological properties of proteins. Chapter 3 focuses on a stability of proteins, including the methods for the stability prediction. Next chapters describe the algorithms, which serve as building blocks of the ancestral sequence reconstruction. In chapter 4, the task of searching homologs of a given sequence is outlined. Chapter 5 is dedicated to the description of a sequence alignment and its algorithms. Phylogenetic trees are covered in chapter 6.

After all the theoretical knowledge, a tool for the automation of ancestral sequence reconstruction called FireProt^{ASR} is introduced in chapter 7. This chapter describes the tool from the architectural and implementation point of view. Novel techniques, which were added into FireProt^{ASR} within this thesis, mainly a successor analysis, are presented. Chapter 8 focuses on the evaluation of the implemented methods and presentation of the new user interface of FireProt^{ASR}.

Chapter 2

Proteins

Proteins are organic macromolecules that play crucial role in the livability of all living organisms. They have a huge variety of functions including the catalysis of chemical reactions, signalization and construction of the robust protein structures. Due to this reason, proteins are a target of an extensive research in fields like biochemistry, proteomics, and bioinformatics. In this chapter, various aspects of protein study are discussed. The formation of proteins is described. A classification of proteins is presented. Last but not least, four levels of protein structure are described.

2.1 Amino acids

Amino acids are basic building blocks of proteins. All amino acids consist of amino (-NH_2) and carboxylate (-COOH) groups and of a side chain (denoted by R). This side chain is different for each amino acid and differentiates between the total number of 20 standard amino acids.

2.1.1 The formation of amino acid chains

Chains of amino acids are called polypeptides. One or more polypeptides creates a protein. In a polypeptide, the amine group of one amino acid reacts with the carboxylic group of the second amino acid and creates a peptide bond. By chaining those peptide bonds, a polypeptide is created. During this polymerization, one molecule of water is yielded for each peptide bond. Amino acids in a polypeptide are also called residues because they are the remainders of the original amino acids after a molecule of water was discarded.

In living organisms, polypeptides are synthesized from template DNA in a process with multiple stages (Figure 2.1).

- **Transcription**: In this phase, template DNA is copied into RNA. The process starts when RNA polymerase binds to a specific place in DNA sequence which marks the start of a coding region. This place is called promoter. RNA polymerase than moves in the direction from 3' to 5' end of the template sequence and adds nucleotides to the newly created RNA sequence. Nucleotides are added based on the compatibility of nucleotide bases. Adenine pairs with uracil and cytosine with guanine. When RNA polymerase reaches DNA sequence called terminator, the transcription is ended. The synthesized RNA sequence is called messenger RNA (mRNA).

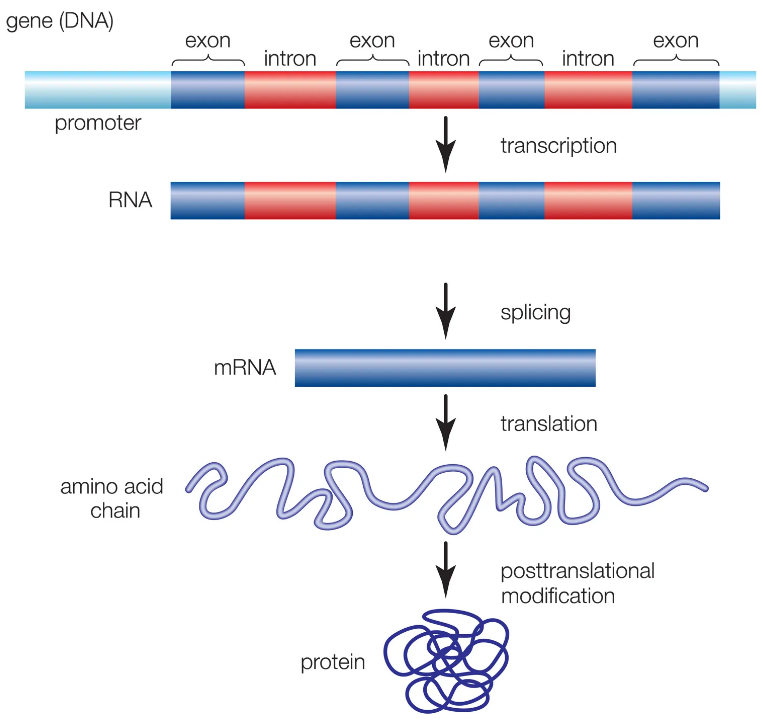


Figure 2.1: The schema of a protein synthesis. Adapted from Britannica[1].

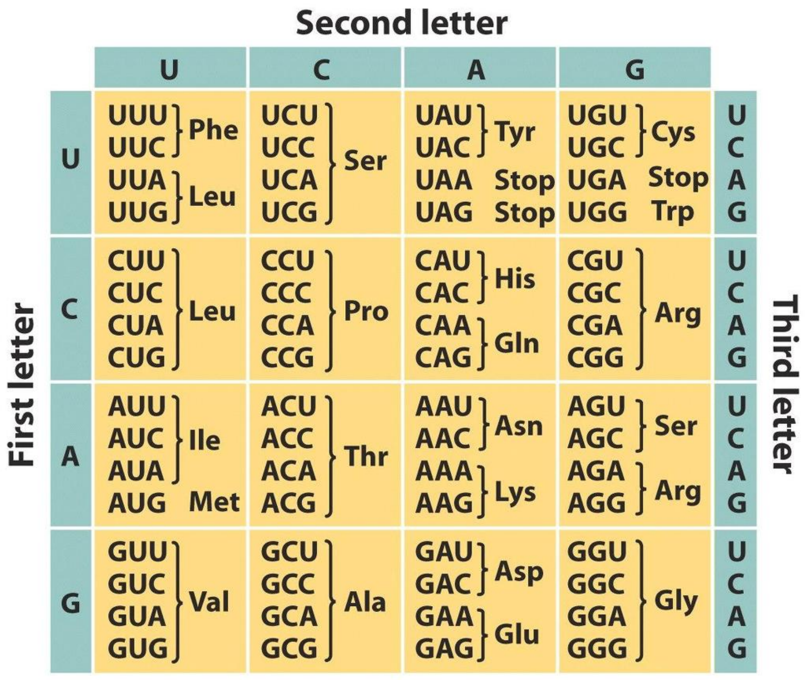
- **Splicing**: This step occurs during protein synthesis only in eukaryotic cells (although in rare cases, splicing was also observed in non-coding RNA in prokaryotes [85]). A lot of eukaryotic genes consist of coding parts (exons) and non-coding parts (introns). During splicing, introns are removed from mRNA, as seen in Figure 2.1.

  An extension of the process of splicing is so-called alternative splicing. Some exons can conditionally be removed from RNA together with introns. It means that one sequence of DNA can code multiple various RNA products (and therefore multiple proteins). For example, one gene of *Drosophila melanogaster* was reported to have 38,016 variants [92].

- **Translation**: The last step is a translation of the mRNA sequence into the amino acid chain. A ribosome binds to mRNA and uses it as a template. It gradually creates a polypeptide by adding amino acids while moving along the mRNA. Each amino acid is coded by a codon consisting of 3 consecutive nucleotides. There are 4 different nucleotide bases, so it creates $4^3 = 64$ combinations. However, there are only 20 standard amino acids. Therefore, the genetic code is told to be degenerate, as some amino acids can be coded by multiple codons. There are also other amino acids, like selenocysteine and pyrrolysine, but those are not included in the genetic code and are coded differently. The genetic code also contains 3 codons which act as stop codons and end the translation. The whole genetic code is shown in Figure 2.2.

2.1.2 Classification of amino acids

There are multiple ways to divide amino acids into categories. One of them is to classify amino acids into hydrophobic and hydrophilic groups. This classification [33] is very impor-



First letter	Second letter: U	C	A	G	Third letter
U	UUU Phe	UCU Ser	UAU Tyr	UGU Cys	U
U	UUC Phe	UCC Ser	UAC Tyr	UGC Cys	C
U	UUA Leu	UCA Ser	UAA Stop	UGA Stop	A
U	UUG Leu	UCG Ser	UAG Stop	UGG Trp	G
C	CUU Leu	CCU Pro	CAU His	CGU Arg	U
C	CUC Leu	CCC Pro	CAC His	CGC Arg	C
C	CUA Leu	CCA Pro	CAA Gln	CGA Arg	A
C	CUG Leu	CCG Pro	CAG Gln	CGG Arg	G
A	AUU Ile	ACU Thr	AAU Asn	AGU Ser	U
A	AUC Ile	ACC Thr	AAC Asn	AGC Ser	C
A	AUA Ile	ACA Thr	AAA Lys	AGA Arg	A
A	AUG Met	ACG Thr	AAG Lys	AGG Arg	G
G	GUU Val	GCU Ala	GAU Asp	GGU Gly	U
G	GUC Val	GCC Ala	GAC Asp	GGC Gly	C
G	GUA Val	GCA Ala	GAA Glu	GGA Gly	A
G	GUG Val	GCG Ala	GAG Glu	GGG Gly	G

Figure 2.2: The genetic code. Adapted from Branden and Tooze[11].

tant because the hydrophobicity and hydrophilicity of residues strongly influence the 3D structure of the protein. In the following overview, each amino acid will be represented by its name and also by a 3-letter and 1-letter symbols which are often used for writing down the protein sequence.

- **Hydrophobic residues**
  - **Aliphatic**: Alanine (Ala, A), Leucine (Leu, L), Isoleucine (Ile, I), Valine (Val, V)
  - **Aromatic**: Phenylalanine (Phe, F), Tyrosine (Tyr, Y), Tryptophan (Trp, W)
  - **Sulfur containing**: Methionine (Met, M), Cysteine (Cys, C)
  - **Hydrogen as a side chain**: Glycine (Gly, G)
- **Hydrophilic residues**
  - **Negative charged**: Aspartic acid (Asp, D), Glutamic acid (Glu, E)
  - **Positive charged**: Histidine (His, H), Lysine (Lys, K), Arginine (Arg, R)
  - **Polar**: Asparagine (Asn, N), Glutamine (Gln, Q), Serine (Ser, S), Threonine (Thr, T), Proline (Pro, P)

AAindex [49] is a database of various properties of amino acids. It contains hundreds of records, such as hydrophobicity or volume values for each amino acid. AAindex also contains pairwise matrices, such as substitution matrices or contact potential matrices.

2.2 Classification of proteins

Proteins are a diverse group of biomolecules. The classification of proteins by their biological function is difficult as the proteins have a vast number of functional roles and this number is continually increasing as the knowledge is deepened. Furthermore, some proteins have multiple functions and therefore, the classification is not that strict. The most important biological functions comprise of: [110]:

- **Enzymes or catalytic proteins**: Enzymes are proteins responsible for the catalysis of chemical reactions. They are crucial for most of the reactions in a cell. Every chemical reaction can occur only when some amount of energy is provided. This minimal required amount of energy is called activation energy. Without enzymes, the activation energy of most chemical reactions in cells would be too high and the reactions would be too slow to produce enough chemical products. Enzymes are able to lower the activation energy (thus accelerate the reaction) by binding the inputs of the reaction. Those molecules are called substrates. In some cases, so-called cofactors have to be present for the catalytic reaction. Cofactors can be either inorganic ions or organic coenzymes.

  Examples of enzymes are RNA polymerase responsible for the transcription of DNA, or pepsin, which is present in a digestive system and helps digest proteins in food by breaking them down into smaller molecules.

- **Contractile proteins**: The most important role of contractile proteins is muscle contraction. Examples are actin and myosin, which are found in all types of muscle cells.

- **Structural or cytoskeletal proteins**: Structural proteins are able to form stiff structures with solid form and shape. For example, collagen forms skin, bones or teeth, and keratin occurs in hair and nails.

- **Transport proteins**: Transport proteins are responsible for moving molecules from one place to another. Membrane transport proteins, such as neurotransmitter transporters, move materials through membranes. Hemoglobin is a protein which occurs in red blood cells and transports oxygen from lungs to the rest of the body.

- **Effector proteins**: The most important representatives of effector proteins are hormones. Nevertheless, not all hormones are proteins. Hormones are a diverse group of molecules, ranging from gasses to proteins. They serve as signaling molecules and they often act in a different organ than where they were produced. Examples of protein hormones are insulin, which is responsible for the regulation of the metabolism, and oxytocin, which plays an important role during reproduction and birth.

- **Defence proteins**: The purpose of defence proteins is to protect the organism. Immunoglobulin, also called antibody, is a crucial part of an immune system. It is able to recognize and neutralize pathogens. Other defence proteins include toxins, such as ricin.

- **Electron transfer proteins**: Electron transfer proteins are able to shuttle an electron from one molecule to another by performing redox reaction [6]. They often contain atoms of metals. For example, ferredoxin contains iron and sulfur and has an important role in photosynthesis.

- **Receptors**: Receptors are proteins that respond to the binding of specific molecules by sending some kind of signal. As an example, acetylcholine receptor identifies acetylcholine, which is one of the most important neurotransmitters.
- **Repressor proteins**: Reprossors regulate chemical reactions and processes in cells. The most prominent example are proteins from Fos and Jun family which regulate the expression of a myriad of genes in a variety of tissues and cell types [13].
- **Chaperones (accessory folding proteins)**: After a protein is synthesised, it has to be folded into a specific 3D conformation. Chaperones help with the correct folding. Many chaperons are heat shock proteins. They are expressed in response to elevated temperature and their purpose is to prevent folding errors caused by heat as the process of protein folding is very sensitive to the temperature [23].
- **Storage proteins**: The purpose of storage proteins is storing various molecules in organism. Proteins gliadin and glutenin are main components of gluten, which is a main storage in wheat grains. Ferritin is a storage of iron. Free iron is toxic, so it has to be bound to the protein. When the iron is needed, ferritin is degraded and the iron is released and used, for example in hemoglobin.

Proteins can also be classified according to the involvement of trace elements [110] as each trace element has some specific use. For example, chlorine participates in maintaining osmotic balance, nickel is found in hydrogenase enzymes and zinc is often a catalytic component of enzymes.

Other important factor for protein classification is a structure of proteins, as seen in the next section.

2.3 Structure of proteins

Protein structure is a very important aspect which has many implications for the function and usage of the proteins. Typically, four levels of structure are distinguished based on their complexity [56], ranging from primary to quaternary, although a supersecondary level and domain level have been proposed between the secondary and tertiary structures [33]. Four main levels of protein structure are [33] (Figure 2.3):

- **Primary structure**: Primary structure is defined as a sequence of amino acids. The protein sequences are often written down as sequences of one-letter amino acid codes, with the direction from amino-terminal to carboxyl-terminal. The primary structure can be easily inferred from a DNA sequence or mRNA sequence with the use of the genetic code table.
- **Secondary structure**: Adjacent amino acids tend to form recurring structure elements, known as secondary structures. They are defined by regular arrangements in protein backbone. Two main motifs of secondary structure are alpha helices and beta sheets. Other motifs, such as omega loops [57], were also discovered, but their properties are not as advantageous as those of alpha helices and beta sheets. Therefore, those motifs are of a rare occurrence in the nature.

  Alpha helix is a structure motif in which each amino acid is bonded to the amino acid four positions further. The hydrogen bonds, which stabilize the helix, are nearly

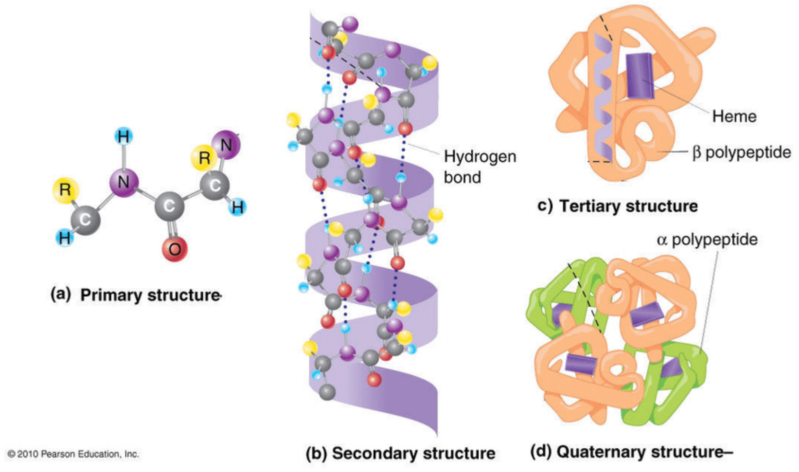


Figure 2.3: Four levels of protein structure. Adapted from Russell[88].

parallel with the long axis of the helix [33]. One turn of alpha helix consists of 3.6 amino acids.

Beta sheet consists of beta strands. A beta strand usually has the length of 3 to 10 amino acids and the backbone of a beta strand is almost fully extended. When multiple beta strands form a beta sheet, amide hydrogens of one beta strand bind with carbonyl oxygens of the second beta strand and create hydrogen bonds.

Not all parts of a protein have to belong to one of the secondary structure motifs. The usual situation is that sections with structure motifs are interrupted by sections without those motifs. These sections are also called random coil [100].

Supersecondary structure is a structure consisting of multiple motifs of a secondary structure. An example of supersecondary structure is helix-turn-helix, which has a function of binding DNA in many proteins.

- **Tertiary structure**: Tertiary structure describes the three dimensional conformation of the polypeptide chain. Tertiary structure of a protein is given by chemical properties of amino acids. After the protein is synthesized, it folds into a tertiary conformation based on those properties. Hydrophobic amino acids tend to hide in a core of the folded protein while hydrophilic amino acids tend to occur at a surface of the protein. This way, amino acids that are very distant in primary structure can get close to each other in a tertiary structure.

  Tertiary structure of a protein can be defined as coordinates of all atoms in a protein and can be determined by a few different methods. The statistics from the biggest database of protein tertiary structures - PDB [8], show that most of the structures was obtained from X-ray crystallography. In this method, a purified sample at high concentration is crystallised and the crystals are exposed to an X-ray beam. The resulting diffraction patterns can be used to determine the coordinates of atoms [102].

  Tertiary structure has many implications for the behavior and usability of the protein. One of the most important processes related to the proteins is a protein-ligand interaction, in which a small molecule called ligand binds a protein. This process is enabled by a high specificity of proteins. For example, the tertiary structure of some proteins contains places which match the shape of a ligand, like lock and key [70].

  A computational prediction of a protein tertiary structure remains an unsolved problem in molecular biology, however, a solution to this problem is extremely important for protein engineering and rational drug design [27]. In recent years, the technology has taken a big step towards the solution of this problem. AlphaFold [44] achieved results comparable with the measurements of experimental methods [96]. However, it still has some drawbacks and is far away from being the perfect solution of the problem.

- **Quaternary structure**: Some proteins consist of multiple polypeptide chains. In that case, the three dimensional conformation of the whole protein is described by a quaternary structure, while the conformation of each polypeptide chain is described by a tertiary structure. The number of polypeptide chains is not correlated to the size of protein. For example, insulin has two chains and its molecular mass is 5700 daltons, while hexokinase has molecular weight of 102 000 daltons, even though it only has one chain [110].

Proteins can be further classified by their structure into three groups [33]:

- **Fibrous proteins**: Fibrous proteins are characterized by long strands of amino acid chains. They are necessary for providing a form and a shape to the organic structures. Examples contain structural and cytoskeletal proteins, such as collagen and keratin.
- **Globular proteins**: Proteins with globular or spherical shape are called globular proteins. According to the prevailing type of secondary structure, they are classified into four groups: all-α, all-β, α+β, and α/β[59]. The difference between the last two is that α+β contains both alpha helices and beta sheets which do not mix along the polypeptide chain, while α/β contains alternating segments of alpha helices and beta sheets. Examples of globular proteins are wide, ranging from hemoglobin to enzymes and hormones.
- **Membrane proteins**: Membrane proteins are an integral part of membranes. They are targets for more than one half of all drugs [79]. For example, receptor proteins and some transport and defence proteins belong to this group.

Chapter 3

Protein stability

The stability of proteins is a very import feature with huge implications into the drug industry. Proteins have evolved in mild conditions of physiological environment [69]. Therefore, they can work best in those conditions. However, it would be beneficial if we could produce proteins that can function even in different conditions. For example, laboratory and industrial environment often has conditions that are too harsh for the native proteins, such as elevated temperature or different pH. The stability of proteins also usually comes with other advantages, such as expression yield [26], half-life [111] and performance in the presence of denaturing agents [81].

Protein stability can be defined as a net balance of forces, which determine whether a protein will be in its native folded conformation or a denatured (unfolded or extended) state [33]. Those forces are dependent on chemical properties of amino acids and the three dimensional arrangement of atoms in the protein. In this chapter, the types of forces will be described together with the mechanism of protein folding. Methods for measuring protein stability will be also presented together with the methods and tools for the computational prediction of protein stability.

3.1 Protein folding

Protein folding is a process in which protein changes its conformation from a random structure created by synthesizing the polypeptide chain into the final folded structure. Cyrus Levinthal stated that this process cannot occur randomly. His co-called Levinthal's paradox [58] shows that even for a small protein of 100 amino acids, only three conformations per residue and a sampling rate on the picosecond scale, it would take more time than the age of universe to fold a protein randomly. In reality, the folding times range from microseconds to seconds, so the logical implication is that the folding is driven by some forces. The thermodynamic hypothesis [7] by Christian Anfinsen states that the tertiary structure of the protein is fully determined by its primary structure (sequence of amino acids), at least for a small globular protein in a standard physiological environment. The thermodynamic hypothesis has a few challenges. For example, it was shown that approximately 0.5–4 % of all proteins switch their fold [82]. Nevertheless, the evidence that the folding is driven by some forces and is not random is indisputable.

The forces that act towards the stability of a protein are covalent and non-covalent interactions. The countering forces are the conformational entropy and nonentropic free energies which contribute to the stability of the unfolded state and therefore act against folding. The net stability of protein is the difference between those two countering types of forces. Interestingly, the net stability is very small (5–10 kcal/mol) compared to the individual forces on both sides [33].

Covalent interactions are characterised by sharing of electron pairs between atoms participating in the bond. Covalent bond is generally stronger than non-covalent interactions. Covalent bonds can be divided into two groups. In polar covalent bonds, the difference between the electronegativity of both atoms in the bond is higher than 0.4. In nonpolar bonds, the difference is less than 0.4. The polarity expresses the fact that polar molecules have negatively and positively charged ends. In proteins, two types of covalent bonds occur: peptide bonds and disulfide bonds.

- **Peptide bond** is a bond between carbon in a carboxyl group of one amino acid and nitrogen in an amino group of the second amino acid. A peptide bond is the strongest type of interaction in proteins. It connects two amino acids and is responsible for creating a primary structure of a protein by forming a polypeptide chain.
- **Disulfide bond**, also known as disulfide bridge, is a bond between two atoms of sulfur. Unlike a peptide bond, a disulfide bond is not a part of a backbone of the polypeptide chain so it does not affect the primary structure. Instead, it stabilizes the ternary structure. There are two amino acids which contain sulfur, methionine and cysteine, but only cysteine is capable of creating disulfide bonds. Therefore, the disulfide bridge always connects two cysteines. Disulfide bond in proteins usually connects two distant places and has an importance for the stability of some proteins, usually proteins secreted to the extracellular medium [97].

Non-covalent interactions are interactions in which atoms do not share electrons. Those interactions are weaker than covalent bonds. Proteins contain multiple types of non-covalent interactions: electrostatic interactions, hydrophobic interactions, hydrogen bonds and van der Waals interactions.

- **Electrostatic interactions** appear between two parts of a protein with a charge. Charges can be either positive or negative and opposite charges attract whereas like charges repel. The main participants in electrostatic interactions are five charged amino acids, lysine, arginine, histidine, aspartic acid and glutamic acid. Those interactions can be strong even at a distance (e.g. of 5–10 Å) [114]. It is also possible to modulate the charges of amino acids, for example by changing pH, which can cause the changes in electrostatic interactions and can lead to a protein denaturation [105].
- **Hydrophobic interaction** is the interaction with the strongest influence on protein folding [33]. Hydrophobic amino acids (e.g. alanine, valine or glycine) have a tendency to exclude water molecules. As a result, those amino acids cluster in the core of globular proteins while the contact between water and non-polar molecules is minimized, which stabilizes the folded protein. Therefore, hydrophobic interaction is the key element determining the tertiary structure of a protein.
- **Hydrogen bond** is a dipole-dipole attraction between an atom of hydrogen, which is partially positively charged, and an atom with high electronegativity, which is partially negatively charged, usually nitrogen, oxygen or fluorine [66]. Hydrogen bonds contribute strongly to the formation of the secondary structure of proteins [33].

- **Van der Waals interactions** are a consequence of quantum mechanics. Electron density in an electron cloud randomly fluctuates. When it temporarily shifts to one side of an atom, it creates a transient charge and other atoms can be attracted to this charge or be repelled by it [63]. Van der Waals forces are weak compared to the other interactions. They depend strongly on the distance of the interacting atoms and they quickly vanish at longer distances.

Conformational entropy is a main force opposing protein folding. It is related directly to the number of different conformations a particular state can adopt [18]. In a denatured unfolded state, the degrees of freedom are higher than in a folded state. In a folded state, the rotation of side chains is restricted as the side chains in the core of a protein generally have only one possible conformation. The common secondary structures, such as alpha helix and beta sheet, also have lower conformational entropy than the random coil.

Type of interaction	Contribution (%)
Hydrophobic	50.8
Hydrogen bonding	27.1
van der Waals	27.1
Electrostatic	6.4
Disulfide	1.1

Table 3.1: Relative contributions of interactions to protein stability [33].

3.2 Measuring the protein stability

In this section, two important aspects of a protein stability measurement are described: the metrics that can be used to quantify the protein stability, and the methods for measuring the stability experimentally in the laboratories.

3.2.1 Metrics of the protein stability

There are multiple metrics to measure the stability of a protein. Gibbs free energy and melting temperature are the two most common. Those two metrics are correlated, but the correspondence is not strictly linear in all conditions [84].

Gibbs free energy is a thermodynamic potential representing the maximum reversible work that may be performed by a thermodynamic system at a constant temperature and pressure. In other words, it is the maximum amount of non-expansion work that can be extracted from a closed system. It is calculated as

$$G = H - TS$$

, where G is the Gibbs free energy (the unit is Joule), H is the enthalpy (the amount of energy stored in thermodynamic system, the unit is Joule), T is the temperature (in Kelvins) and S is the entropy (the unit is Joule per Kelvin). Every thermodynamic system has a tendency to reach a state with the minimal Gibbs free energy. Therefore, by comparing the Gibbs free energy for unfolded and folded protein, it is possible to determine which of those two conformation will be preferred by the protein. Furthermore, the stability of a protein can be quantified as a change of Gibbs free energy upon folding. This change is denoted as ΔG and is calculated as

$$\Delta G = G_{folded} - G_{unfolded}$$

. Negative value of ΔG indicates that the folded state has lower Gibbs free energy and is therefore more stable.

Melting temperature is the second metric used for quantifying the protein stability. The melting temperature (T_m) is defined as a temperature in which half of the population of proteins is folded and the other half is unfolded. This happens when the Gibbs free energies of the folded and unfolded states are equal.

$$\Delta G(T_m) = 0$$

Higher value of melting temperature means that the protein is more stable because it can remain folded even in harsher conditions of the elevated temperature.

3.2.2 Methods for measuring the protein stability

There are a plenty of methods for measuring the stability of a protein. Three important methods will be presented in this section: differential scanning calorimetry, circular dichroism spectroscopy and fluorescence-based assays.

Differential scanning calorimetry (DSC) is one of the most widely used tools for studying the thermodynamics of a protein unfolding. In DSC, heat capacity of a protein in a solution is directly measured versus temperature [33]. Furthermore, the characteristics of a sample are compared with a reference sample with well-known heat capacity. DSC can be used to measure Gibbs free energy at various temperatures.

Circular dichroism (CD) spectroscopy measures the differences between the absorption of left-hand and right-hand circular polarized light. This way, it can detect a structural asymmetry of a protein. The ultraviolet CD spectroscopy is a great tool for the estimation of the secondary structure [32]. The CD spectroscopy is not as accurate as some other methods, however, it is an easy method which does not require large amount of a protein. In result, it can be used to quickly examine the protein in various conditions.

Fluorescence-based assays use the fluorescence of tryptophan and tyrosine residues during a protein unfolding. Tryptophan residues provide the most valuable information because their fluorescence is very sensitive to their environment. The advantage of fluorescence-based techniques is that only a very small amount of a protein is required [21].

3.3 Methods for the protein stability prediction

The laboratory experiments are costly and the number of various proteins is huge. Therefore, it is clear that obtaining the data about a protein stability only through the laboratory measurements is not very practical. In some cases, it is necessary to quickly investigate the stability of many proteins, ideally without the tedious preparation and expensive manual experiments. Naturally, computer simulations are used for this task. In this section, four categories of the *in silico* methods for the protein stability prediction will be presented: force-field methods, methods based on phylogenetic analysis, machine learning and combined predictors [72].

3.3.1 Force-field methods

Force-field methods use a simplified quantum mechanic principles to evaluate the proteins free energy. The advantage of this method is that it does not need any training data. The whole prediction is based solely on the physical and chemical forces in the molecule. One example of a force field can be field created from the interactions mentioned in section 3.1. In such a case, the Gibbs free energy of the folded state can be computed as a sum of all partial free energies [33]:

$$G_F = G_{hy} + G_{el} + G_{hb} + G_{vw} + G_{ss}$$

, where G_{hy}, G_{el}, G_{hb}, G_{vw}, and G_{ss} are, respectively, hydrophobic, electrostatic, hydrogen bonding, van der Waals, and disulfide bonding free energies, and G_F is the resulting Gibbs free energy of the folded protein. The Gibbs free energy of the unfolded state is computed as [33]

$$G_U = G_{en} + G_{ne}$$

, where G_{en} is the entropic free energy and G_{ne} is the non-entropic free energy. G_U is the free energy of the unfolded state. It was shown that this free energy increases with the size of the protein. Each residue adds approximately 1.2 kcal/mol [48]. Furthermore, there are some hydrogen bonds even in the unfolded state. The number of hydrogen bonds is approximated as a half of the number of hydrogen bonds in a folded protein. The free energy of the unfolded protein can thus be calculated as [33]

$$G_U = (1.2N + (1/2)G_{hb})\ kcal/mol$$

, where N is the number of residues in the protein.

Force-field methods are very accurate, but also computationally expensive. For more complex proteins, a heuristic version of those methods often has to be used.

3.3.2 Methods based on phylogenetic analysis

The phylogenetic analysis uses the data from a set of homologous sequences. It stands on the presumption that highly conserved amino acids have an important role for the stability of a protein. Those amino acids were selected in the process of evolution and should therefore be beneficial. The main advantage of those methods is that they require only the data about the sequences of the proteins. In the era of the next generation sequencing, the amount of such a data is huge. Two main methods based on the phylogenetic analysis are consensus design and ancestral sequence reconstruction [72].

In **consensus design**, the multiple sequence alignment of a set of homologous sequences is created. The most frequent amino acids on each position are called consensus residues and should represent the stabilizing amino acids. Consensus design is very sensitive to the initial selection of the homologous sequences. When the sequences are very diverse, the method has a problem with preserving the function of the protein. This problem is strongest when both prokaryotic and eukaryotic sequences are used [41]. On the other hand, when the multiple sequence alignment is dominated by only a few subfamilies, a phylogenetic bias can occur [55].

Ancestral sequence reconstruction (ASR) is a method which reconstructs the ancestral proteins from a set of homologous sequences. The ancestral sequences often have higher stability, possibly because of the thermophilic origin of primordial life [2]. ASR is a complex method with multiple steps, including the construction of a phylogenetic tree, and will be discussed in detail in the section 6.4.

3.3.3 Machine learning

In recent years, machine learning methods have been going through a massive increase in popularity. They have proven to be competitive with the state-of-the-art methods in many fields including bioinformatics. Machine learning uses statistical techniques to recognize patterns in a set of examples. The biggest strength is the ability to find complex relationships hidden in the data. Such patterns are often unrecognizable even by the experts in the field. Another advantage is that the system does not have to be constructed from scratch for each task as the machine learning systems are constructed from reusable blocks. The specialization for specific task is achieved by training the system with input data specific for the task. Therefore, successful machine learning systems can be created even by non-experts in the given field.

The crucial factor of machine learning methods is the availability of a sufficient amount of data with high quality. Ideally, the data should be consistent, reliable, diverse and balanced. In reality, the datasets with such qualities are rare. For example, ProTherm [54], the biggest database of protein thermodynamic measurements, contains a lot of redundancy and inaccurate information [65]. It is possible to use only a subset of ProTherm, such as PoPMuSiC dataset [17], however, the small size of the reduced dataset can be problematic. The problem is particularly significant for neural networks and deep learning. These methods can recognize immensely complex patterns but generalize poorly when the amount of training examples is not sufficient. The most successful machine learning tools for the stability prediction are based on the support vector machines and random forests [72].

Another problematic aspect of machine learning is the validation of results. In order to obtain unbiased results, tools should be validated on inputs that were not used for training. However, k-fold cross-validation is often used instead. It means that the training data is reused for the validation. This approach is useful when the size of dataset is small, but it can also lead to an inaccurate validation, namely to an overestimation of the accuracy of the tool [83]. Furthermore, tools trained on one dataset can generalize badly on other datasets.

3.3.4 Hybrid approaches

Methods based on hybrid approach use the combination of multiple methods. By considering various different methods, they try to overcome the drawbacks of each particular approach. The predictions of several methods can be combined either by majority voting or by some root classifier which assigns weights to all subsystems. Advanced techniques can use more complex workflow instead of simply summing up the predictions of the subsystems. Some tools, such as FireProt [73], are able to benefit from the knowledge of the conserved regions and active sites. Such tools can be used to design thermostable mutant proteins, even with multi-point mutations. The stability prediction for multi-point mutants is a hard task because the effect of the combination of mutations is not just a sum of individual effects. Hybrid approaches can mitigate the risk of an inaccurate prediction is such cases.

Chapter 4

Homolog search

Sequence homolog search is one of the most common tasks in bioinformatics. The main goal is to search the existing database and find the most similar sequences to a query sequence. The desire is often to find sequences with a common ancestry, so called homologs. The results can be used to classify the unknown sequence. Furthermore, homolog search is an important part of the phylogeny based protein stability prediction methods (section 3.3.2). Those methods work with multiple sequence alignment of a set of homologous sequences. The initial selection of those sequences is crucial and has tremendous impact on the accuracy of those methods. In this chapter, the concepts related to homologous sequences will be introduced, together with the databases of sequences. Furthermore, useful methods and techniques of the homolog search will be presented.

4.1 Homologs

Homologs are sequences with shared ancestry. Homologous sequences are derived from a common ancestor sequence so the sequence homology is often connected with a sequence similarity. However, each homolog evolved independently, and therefore, even the sequences with a low sequence similarity can be homologs. This is particularly true when the homologs were separated a long time ago and underwent a lot of mutations. Homologous sequences can be divided into two groups: orthologs and paralogs [53]. Their difference is shown in Figure 4.1.

- **Orthologs** are sequences created in the process of speciation. When one species diverges into multiple species, they all have a copy of the homologous sequence in their genome.

- **Paralogs** are created by a sequence duplication. In this process, one gene can be copied into the different positions in genome. As a result, the organism has multiple copies of the same sequence. When the sequence duplication takes place in the ancestor before the speciation, the sequences are called outparalogs. On the other hand, inparalogs are created in one species without the speciation [53].

Majority of homologous sequences can be classified into those two groups. However, a handful of new categories have been created recently to further classify homologs. For example, xenologs are homologs resulting from the horizontal gene transfer [53].

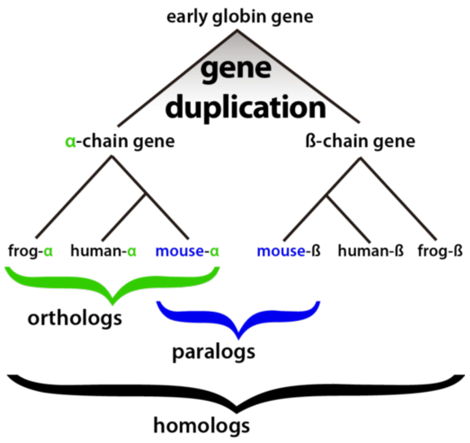


Figure 4.1: Types of homologs shown on an example of globin genes. Adapted from Nagar[75].

4.2 Databases of relevant sequences

The search for homologous sequences requires a database in which the sequences are searched. The database should be sufficiently large and contain all relevant homologs. However, the size of the database is not the only concern. The methods of next-generation sequencing (NGS) are able to produce huge amounts of the sequenation data. The problem is that this data is often redundant and does not have sufficient quality. Some databases ignore this problem and focus rather on the quantity of data available for the scientific community. On the other hand, there are also a plenty of databases which try to provide a filtered subset of sequences with specific qualities. Last but not least, there are also databases with narrow specialization, such as databases of orthologous genes. This section brings a selection of the databases of biological sequences. It is not an exhaustive list, as there are hundreds of such databases, but rather a small selection of the important ones.

4.2.1 UniProt

UniProt [108] is the largest freely accessible database of protein sequences. In 2021, it contained over 225 million sequences. The most important part of UniProt is UniProtKB (UniProt Knowledgebase) which consists of two databases: Swiss-Prot and TrEMBL.

Swiss-Prot is a database of manually annotated sequences. The source of sequences for Swiss-Prot is scientific literature and a curator-evaluated computational analysis. The manual process secures high reliability of the sequences and their annotations, however, it is also laborious and time-consuming. Only a small fraction of the UniProt records is stored in the Swiss-Prot database. In 2021, there were less than 600 thousand sequences, which makes only about 0.3 % of UniProt.

TrEMBL is a database of computationally analyzed sequences. Most of the sequences in TrEMBL were created by the translation of annotated coding sequences from databases of nucleotide base sequences, mainly from the International nucleotide sequence database collaboration (INSDC) [46], which connects databases ENA, GenBank and DDBJ.

4.2.2 Non-redundant (NR) sequence database

With the next-generation sequencing, the amount of sequences in databases grows exponentially. Databases without manual annotations contain a lot of redundant records as one sequence can be independently sequenced by multiple laboratories around the world. All the versions of the sequence can be identical, but minor differences often occur. All those sequences are then added into the database and the redundancy is created. The redundancy negatively impacts the performance of operations with the database such as search for homologous sequences. The redundant sequences do not add any informational value while slowing the system down. The other disadvantage is that the redundancy has a negative effect on some algorithms. For example, ancestral sequence reconstruction requires a set of homologous sequences. When the sequences are identical or very similar, the results of the method are worse than with diverse sequences. The database of non-redundant sequences (NR) [14] was created to solve those problems. In NR, every sequence is a representative of a cluster of similar sequences and has maximal similarity with other sequences in NR lower than some threshold. 90 % is a standard value of such threshold. The non-redundant database is a default database used for performing BLAST similarity searches provided by National Center for Biotechnology Information (NCBI) [14].

4.2.3 Databases of homologous gene families

HOVERGEN and HOGENOM [80] are two databases of homologous gene families. HOVERGEN contains sequences from vertebrate genomes while HOGENOM focuses on bacteria, archaea and unicellular eukaryotes. The main source of data is UniProt [108]. Selected sequences from UniProt are clustered together to assemble protein families. After that, MUSCLE [19] is used to compute multiple sequence alignment (MSA) on the sequences. As a last step, phylogenetic trees are constructed from the MSA with PhyML [35].

The inherent availability of the MSAs and phylogenetic trees is the main advantage of HOVERGEN and HOGENOM. Unlike traditional databases of sequences, HOVERGEN and HOGENOM allow users to perform tree-pattern based searches allowing, among other uses, to retrieve sets of orthologous genes. However, the database is not updated anymore so it does not contain any sequences added to UniProt in recent years.

4.3 Residue conservation

As mentioned previously, the homologous sequences have some level of sequence identity, but this level can be fairly low. It was shown that the sequence identity of 30 % implies 90 % probability that the sequences are homologous [87]. As the identity falls to 20 %, the probability of the homology decreases rapidly, but there are still homologs with only 20 % identity. At the same time, a lot of homologs share the same functional properties. This is not generally true for all homologs as the function shifted during the evolution in some cases, but it is very common, especially in cases with higher sequence identity. Consequently, there must exist some mechanism which ensures that the protein preserves its function during the evolution even when the sequence is mutated.

Residue conservation is a phenomenon which leads to the different levels of mutation on each position in the homologous sequences. In particular, the residue is said to be conserved when there is a low diversity on a specific position among a set of homologs. In such case, most of the rows of the MSA contain the same amino acid in the column which represents the conserved position. On the other hand, high diversity in the column means that the position is not conserved and mutated frequently during the evolution. Although the residue conservation can be examined just by looking at the MSA, the exact scoring of the residue conservation is not standardized [109].

The study of conserved residues can reveal a lot of interesting information about the protein. The main idea is that conserved residues are important for the functionality of the protein and therefore withstood the long periods of time without mutating. A mutation of a conserved residue usually brings an evolutionary disadvantage and therefore is not viable due to the natural selection [15]. Conserved residues are often important for the protein stability. They also occur in clusters in active sites where the protein interacts with other molecules [34]. However, the analysis of conserved residues is beneficial not only for the active sites prediction. It can be also used for the prediction of the tertiary structure because the residues occurring together in an active site in the tertiary structure are often distant in the sequence. The correlation of mutations in those distant residues can be an evidence that they are close to each other in the tertiary structure [31].

When talking about a residue conservation, a similarity of residues is often considered instead of the identity. For example, the mutation of one hydrophobic amino acid to another hydrophobic amino acid does not have as strong effect on the stability of the protein as the mutation to hydrophilic amino acid [68].

4.4 Search algorithms

There are a plenty of algorithms for searching homologous sequences in the sequence database. They are based on the pairwise alignment between the query sequence and sequences in the database. The algorithms can be divided into two groups. Classical methods guarantee that the best possible solution is found, so they satisfy a so-called optimality. On the other hand, the result found by heuristic approaches is not guaranteed to be optimal. However, the big advantage of heuristic algorithms is the speed. They are computationally much more efficient while the precision of results is often only slightly worse than in the case of the classical methods. Thus, all search algorithms used in practice are heuristic algorithms.

4.4.1 BLAST

BLAST [4] (Basic Local Alignment Search Tool) is one of the most important algorithms in bioinformatics. In 1990, a method for calculating a statistical significance of sequence features was presented [45]. Based on those findings, BLAST was created in the same year. The algorithm has three main phases:

- **Seeding**: At the beginning, the repeats and low-complexity regions are removed from the query sequence because they are not relevant for the search. As a next step, a list of words (w-mers) is assembled from the query sequence with the sliding window technique. For proteins, the word length is usually 3. For each word, a set of alternative words is created by the substitutions of individual letters. The similarity score with the original word is computed for each alternative word with the help of a scoring matrix, for example BLOSUM62 [37]. Alternative words with the score higher than a threshold T are called high-scoring words and are used in the rest of the algorithm, other alternative words are discarded. The database is than scanned for exact matches with the high-scoring words.

- **Extension**: In this phase, hits from the previous step are extended in both directions. The high-scoring segment pair (HSP) score of the extended regions is computed continuously. When the score starts to decrease, the extension is ended.
- **Evaluation**: For regions with HSP score higher than a threshold S, the statistical significance (e-value) is calculated. When there is a group of multiple consistent HSP regions, they are all merged together into bigger region. Regions with enough significance are reported to the output.

BLAST is one of the essential bioinformatical tools and has many different versions. Two most important programs are BLASTn and BLASTp for searches involving sequences of nucleotides and proteins, respectively. There are versions working with the transcription of nucleotides into a protein, versions with specific hardware accelerations or parallel versions of BLAST. Also, many more recent heuristic searching algorithms are inspired by the ideas presented by the authors of BLAST.

4.4.2 FASTA

FASTA [60] is one of the first heuristic algorithms for a sequence comparison. It was developed in 1985, five years before BLAST. In some aspects, it is similar to BLAST. In the seeding phase, FASTA does not create alternative words. It only searches for exact matches instead. FASTA also usually uses smaller word size – 2 for proteins. As a consequence, FASTA is more sensitive than BLAST and it can detect more distant homologs. On the other hand, FASTA is not as fast as BLAST, and it also does not remove the low complexity regions, which can have negative impact on the quality of results.

4.4.3 PSI-BLAST

PSI-BLAST [5] (Position-Specific Iterative BLAST) is an important improvement of BLAST. It was developed in 1997 and one of its authors, David J. Lipman, is also a co-author of FASTA and the original BLAST. The idea behind PSI-BLAST is that the position-specific score matrices, also called profiles or motifs, provide better results for homolog searches. PSI-BLAST is able to create those profiles in an automated way by iteratively applying BLAST. In the first step, regular BLAST search is performed. A profile sequence is then created from the result. This profile sequence represents all the found sequences and their important features. The profile sequence is used as a query in the next iteration of the algorithm. In each iteration, more distant homologs are found. Although the method is slower than BLAST because it runs multiple iterations of BLAST searches, the ability to find distant homologs is a very valuable advantage of PSI-BLAST.

4.4.4 BLAT

BLAT [50] (BLAST-like alignment tool) was developed to provide extremely fast alignment and homolog search. BLAT achieves a fast searching speed by creating an index of non-overlapping words (w-mers) and their positions in the sequence database. Words which occur too often are excluded from the index. It also builds a list of all (overlapping) words of the query sequence. It searches the words from the query in the index and creates a list of hits. Multiple consistent hits are merged together. BLAT is not able to detect as distant homologs as BLAST, FASTA or PSI-BLAST, however, it achieves good results in much shorter time than other tools for the set of similar sequences.

4.4.5 HMMER

HMMER [28] uses an MSA to create hidden Markov model of a profile of the sequences. The utilization of a sequence profile resembles PSI-BLAST, but the novelty of HMMER is in the usage of hidden Markov models. HMMER is able to find distant homologs as it detects the underlying patterns in the sequences. The latest version, HMMER3, brought performance improvements and HMMER3's speed is comparable to the speed of BLAST while recognizing more distant homologs than BLAST.

Chapter 5

Sequence alignment

Sequence alignment is an essential task in bioinformatics and has countless usages. It is important for phylogenetic analysis in which the evolutionary development of organisms is studied. For example, phylogenetic trees are constructed with the help of a sequence alignment. It is also necessary for searching homolog sequences in large databases of biological data. Mapping sequences to the reference genome and mutation detection are just other tasks which stand on sequence alignment. Next generation sequencing (NGS) techniques also heavily depend on algorithms for sequence alignment. It can be also used for the prediction of the protein tertiary structure. Last but not least, sequence alignment is extensively used in metagenomics.

Sequence alignment is a quite general task and can be applied even to problems unrelated to bioinformatics, such as natural language processing. Its goal is to find a one-to-one correspondence between the letters of two or more sequences [33]. In the bioinformatics context, the sequences are proteins or DNA and RNA sequences. Therefore, the correspondence is established between individual amino acids or nucleobases, respectively. This chapter focuses primarily on protein sequences, but the algorithms are generally applicable also for the DNA and RNA sequences.

5.1 Alignment scoring

One of the simplest methods for sequence alignment is a dot plot [30]. It is a two-dimensional matrix where rows represent one sequence and columns the second sequence. Each cell contains a dot if the sequences on corresponding positions are equal, otherwise the cell is empty. Matching sections of the sequences can be found as diagonal lines in the plot. An example of a box plot is shown in Figure 5.1. Nevertheless, a dot plot is only a graphical tool. It can be useful for fast visual inspection of the possible alignments, but it cannot be used for automatized processing of data.

Dot plot should serve as an example that, in practise, sequence alignment algorithms should work with some numeric evaluations of the quality of the alignments. This allows them to compare various alternative alignments and choose the best one. However, the design of the scoring function is not straightforward and there are multiple approaches which correspond with the specific needs of various bioinformatics tasks. All scoring functions consider three possible situations on each position in the alignment:

- **Match**: identity of amino acids on some position of the alignment means that this position contributes positively to the overall score.

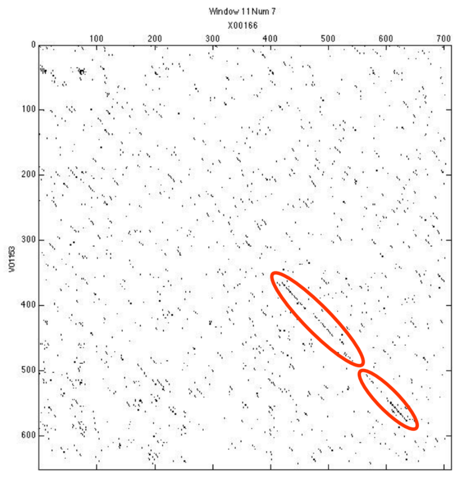


Figure 5.1: An example of a dot plot. Adapted from Martínek[64].

- **Substitution**: the amino acids differ. Substitutions are relatively common in the evolution. The score of the position with a substitution usually depends on particular amino acids.
- **Insertion/deletion (indel)**: there is a gap in one sequence which signalizes a deletion of amino acid in that sequence (or alternatively an insertion in the second sequence). Indels are not as common as substitutions so the contribution to the overall score should be more negative in the case of indels.

5.1.1 Types of alignment

There are three types of sequence alignment which differ in a way they deal with gaps at the beginning and end of the alignment.

- **Global alignment** penalizes all gaps, even at the beginning and the end of the alignment. It is well suited for the alignment of sequences of a similar length. For example, homologous proteins can be aligned globally.
- **Semi-global alignment** solves the problem when one sequence is significantly shorter than the other. Semi-global alignment does not penalize gaps at the beginning and end. Instead, the goal is to find the shorter sequence in the longer sequence. For example, semi-global alignment is used when searching the location of a gene in the genome.
- **Local alignment** does not align whole sequences, but only their subsequences. Local alignment can be used even for dissimilar sequences. It can be used to detect conserved regions or motifs in proteins. Local alignment is also able to find multiple occurrences of the shorter sequences in the longer one.

5.1.2 Substitution matrices

In a simple case, the score of an amino acid substitution is unified and does not depend on individual amino acids involved in the substitution. However, this approach does not reflect the evolutionary processes which lead to mutations. In reality, some mutations occur more often than others. The classification of amino acids was already mentioned in section 2.1.2. The substitutions between similar amino acids (for example leucine and isoleucine) are more probable as such substitutions usually preserve the function of the protein. On the other hand, mutation between dissimilar amino acids (for example hydrophobic alanine and hydrophilic proline) can lead to a defective protein. In order to incorporate these differences into the process of alignment scoring, substitution matrices were created.

An amino acid substitution matrix has one row and one column for each amino acid. Each cell contains a score for the substitution of the corresponding amino acids. The values in the substitution matrices are usually derived stochastically by observing the mutations in a huge number of sequences. This way, it is possible to estimate the probabilities of individual substitutions. The two most important substitution matrices are PAM [16] and BLOSUM [38]. PAM matrices are better for closely related proteins whereas BLOSUM is more suited for distant sequences. Both matrices have multiple variants for various sequence distances. PAM250 and BLOSUM62 are the most common ones.

5.1.3 Gap penalty

Another component of the alignment scoring is a gap penalty. In a simple case, each position with a gap has the same score. This score should presumably be negative, or alternatively, the gap penalty should be positive. But from the evolutionary point of view, few longer gaps are much more likely to occur than a lot of small gaps. It would be possible to use constant scoring and assign the same score to each gap, no matter how long it is, but it would be again too oversimplified. The widespread compromise between these two approaches is an affine gap scoring [3]. It works with two kinds of gap penalties. One is a gap opening penalty, which penalizes the existence of the gap, and the second is a gap extension penalty, which represents the length of the gap. For example, in BLASTp (BLAST for proteins), the default values are 11 for the gap opening penalty and 1 for the gap extension penalty.

5.2 Pairwise alignment

Pairwise sequence alignment is a process whose goal is to align two sequences. These sequences can be aligned globally, semi-globally or locally. The naive algorithm creates all possible alignments of two sequences, counts the score for all alignments and chooses the alignment with the highest score. However, the number of possible alignments grows extremely fast with the length of sequences. For sequences with lengths 100 and 95, there are approximately 55 million alignments. An elegant way to reduce the size of the search space is dynamic programming. Dynamic programming is a method used in various fields. It recursively breaks down a problem into sub-problems. There is often a lot of identical sub-problems, which can be solved only once and the result can be reused. This approach can dramatically speed up the computation. The most prominent algorithms for pairwise alignment, apart from heuristic algorithms, are Needleman–Wunsch and Smith–Waterman algorithms, which both use dynamic programming.

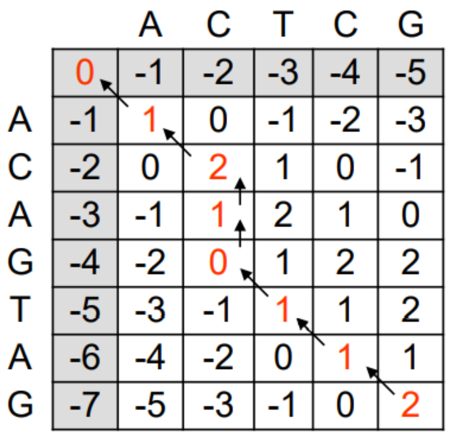
5.2.1 Needleman–Wunsch algorithm

Needleman–Wunsch algorithm [76], developed in 1970, is an algorithm for global sequence alignment. It is one of the most influential algorithms in bioinformatics as it pioneered the usage of dynamic programming for bioinformatics problems. The idea is that in the first step, there are three options – it is possible either to align the first characters or insert a gap into the first sequence or into the second sequence. The alignment of the remaining parts can be again recursively broken down into similar three options. This approach creates a 3-ary tree of possible states, in which certain states occur multiple times. It can be shown that the tree can be rewritten as a 2D matrix. Rows and columns represent characters of the first and the second sequence, respectively. An example of such table is shown in Figure 5.2. The algorithm has two steps:

- **Filling in the table**: the cell in the top-left corner has a score 0. The cells are traversed row by row. For each cell, the score is computed with the use of the neighboring cells as a maximum of three values:
  - The score of the top cell + gap penalization
  - The score of the left cell + gap penalization
  - The score of the top-left diagonal cell + the score for the combination of the row and column characters, which is obtained with respect to the chosen substitution matrix.

  This way, all cells are filled. The value in the bottom-right corner represents the score of the optimal alignment.

- **Tracing back the optimal alignment**: the matrix is traversed from the bottom-right corner to the top-left corner. In each cell, the path goes to the cell which was previously used for calculating the value of the actual cell. The resulting path represents the optimal alignment:
  - The diagonal move represents a match or mismatch
  - Vertical move represents a gap in the horizontal sequence
  - Horizontal move represents a gap in the vertical sequence



		A	C	T	C	G
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
A	-3	-1	1	2	1	0
G	-4	-2	0	1	2	2
T	-5	-3	-1	1	1	2
A	-6	-4	-2	0	1	1
G	-7	-5	-3	-1	0	2

Figure 5.2: An example of the second step of Needleman–Wunsch algorithm. Adapted from Martínek[64].

5.2.2 Smith–Waterman algorithm

Smith–Waterman algorithm [101] is another important algorithm which uses dynamic programming. It is a variation of Needleman–Wunsch, however, it is used for the local alignment instead of the global alignment. Both algorithms coincide in the usage of a 2D matrix. In the first step, the matrix is filled in respecting the same rules as in the case of Needleman–Wunsch. However, there is an important adjustment – negative scores are not allowed, so if a cell would have a negative value, 0 is used instead. In the second step of the algorithm, the tracing back does not start in the bottom-right corner, but in the cell with the highest score. Such score represents the optimal local alignment. Multiple cells with the highest score mean that there are multiple optimal alignments. The tracing back continues until a cell with 0 is reached. The traversed path represents the particular alignment.

5.3 Multiple sequence alignment

Pairwise alignment is an irreplaceable tool for searching sequences in databases, and many other tasks. However, in some other cases, it would be beneficial to align more than two sequences. For example, phylogenetic trees are constructed from the alignment of multiple sequences. Additionally, a multiple sequence alignment (MSA) of the similar sequences can even help improve the pairwise alignment. The algorithms presented in the previous section only work with two sequences. It is possible to extend them so that they are applicable with multiple sequences, even though the complexity grows exponentially with the number of sequences. Such algorithms cannot be used in real-world applications. Instead, two types of heuristic algorithms are used: progressive and iterative methods.

5.3.1 Progressive methods

Progressive methods work in three steps:

- Pairwise distances between all pairs of sequences are calculated. The sequences are then iteratively clustered and a rooted tree is created, based on the clusters. This so-called guided tree serves as a guide for the following steps of the algorithm.
- Two closest sequences are chosen with the respect to the guided tree. They are aligned and the number of sequences decreases from k to $k - 1$.
- The previous step is repeated until all sequences are aligned in one alignment.

Progressive methods have multiple disadvantages. The first problem is their speed, which is mainly limited by the first step of the algorithm. In order to construct the guided tree, it is necessary to calculate all pairwise distances between the sequences. The number of pairs grows quadratically with the number of sequences. This approach is unfeasible in practise when the pairwise alignments are computed with exact methods using dynamic programming. Nevertheless, the exact pairwise alignment is not needed. The guided tree construction only requires the information about pairwise distances between sequences. Those distances can be computed with the help of a heuristic method. The k-mers are created from the sequences, and a vector of k-mer frequencies is computed for each sequence. The distance between the sequences is then calculated as the distance between these vectors.

The second problem of progressive methods is that they depend heavily on the quality of the guided tree created in the initial step. Every error from earlier stages of the algorithm propagates into the final result. This problem can be solved by iterative methods.

The most important progressive methods belong to the Clustal family [12]. The first version, named simply Clustal, implements the three-step method described earlier. For computing pairwise distances, k-mer counting (k is 1 or 2 for proteins) is used. The guided tree is constructed via an UPGMA [103] (unweighted pair group method with arithmetic mean) method. ClustalW [106] adds sequence weights. It introduces also other improvements, such as using a neighbor-joining method [89] for the guided tree construction, or a dynamic selection of the scoring matrix. ClustalΩ[99] uses hidden Markov models and is well suited even for the large sets of sequences.

Apart from Clustal, progressive methods include MAFFT [47], which uses fast Fourier transformation for the sequence clustering, or T-Coffee [78].

5.3.2 Iterative methods

One of the main problems of progressive methods is that while the initial alignment is expanded, there is no way to correct the errors and every error is propagated into the final alignment. The iterative methods are an answer to this issue. The advantage of iterative methods is that they iteratively refine the alignment. It means that the sub-MSAs created during the algorithm run are not resolute and can be changed in the next iterations.

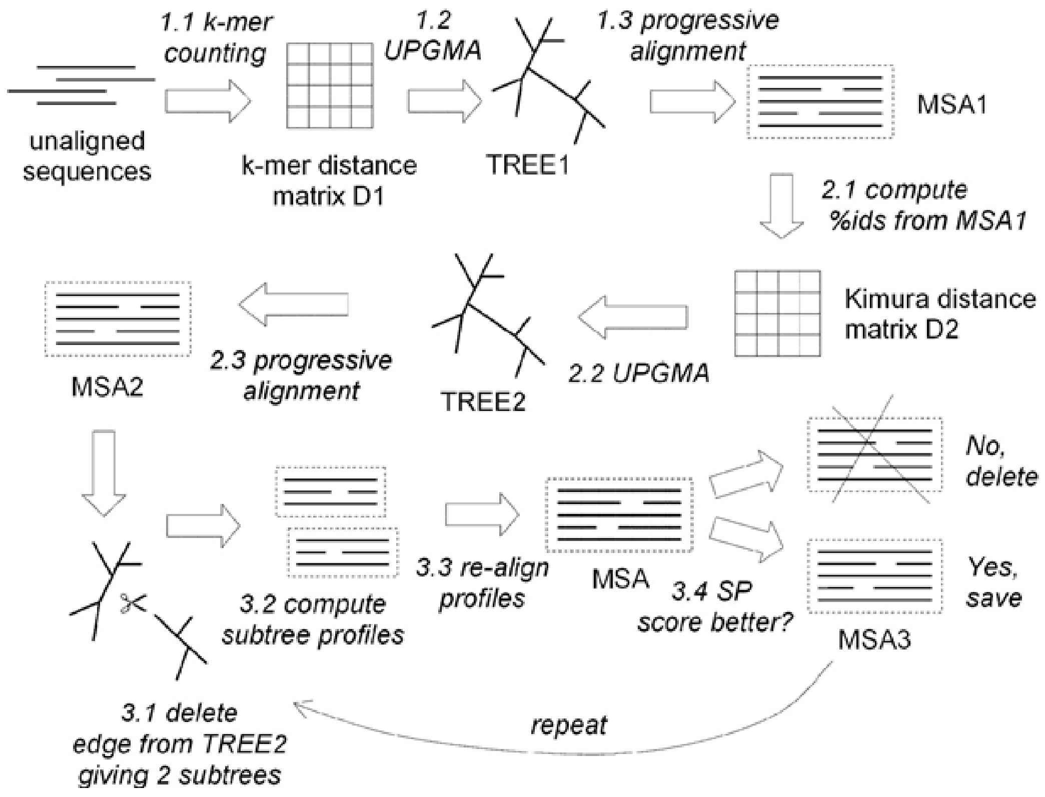

1.1 k-mer counting
1.2 UPGMA
1.3 progressive alignment
unaligned sequences
k-mer distance matrix D1
TREE1
MSA1
2.1 compute %ids from MSA1
Kimura distance matrix D2
2.2 UPGMA
TREE2
2.3 progressive alignment
MSA2
3.1 delete edge from TREE2 giving 2 subtrees
3.2 compute subtree profiles
3.3 re-align profiles
MSA
3.4 SP score better?
No, delete
Yes, save
MSA3
repeat

Figure 5.3: A schema of MUSCLE. Adapted from Edgar[19].

The most significant iterative MSA algorithm is MUSCLE [19]. The algorithm is schematically shown in Figure 5.3 and has three stages:

- In the **draft progressive stage**, the algorithm resembles progressive methods. Pairwise distances are computed using k-mers. The distances are used to build a guided tree using UPGMA method, and the alignment is constructed with the help of the tree.
- In the **improved progressive stage**, an improved alignment is created. The original paper of MUSCLE states that the main source of error in the draft progressive stage is the approximate k-mer distance measure. To fix this error, a more accurate Kimura distance[51] is used in the second stage. The Kimura distances are computed using the alignment created in the first stage. Again, the distances are used to create a tree by UPGMA method, and the tree serves as a guide for the improved alignment.
- **Refinement stage** works in multiple iterations. In each iteration, one edge is removed from the guided tree. This way, two subtrees are created. Profiles for both subtrees are computed and a new alignment is created by aligning these two profiles. If the new alignment is better than the previous one, it is used as an input for the next iteration.

Chapter 6

Phylogenetic trees

The phylogenetic analysis is a powerful method used extensively in bioinformatics. It reflects the evolutionary relationships between a set of organisms or sequences. This way, it is able to reveal an interesting information about the organisms. It is the key analysis for the taxonomic classification of species. It is also useful for a protein engineering, for example, in the analysis of the conserved regions.

These days, the phylogenetic analysis and the construction of phylogenetic trees is closely connected with the analysis of biological sequences, whether DNA, RNA, or proteins. However, the concept of the phylogenetic trees is much older than the discovery of those sequences as the phylogenetic tree represents an intuitive visualization of the taxonomic relations between species during the evolution. In this chapter, the definition of the phylogenetic tree is presented together with the methods for the construction of the phylogenetic trees.

6.1 Phylogenetic tree

Phylogenetic tree is a rooted or unrooted tree. The leaf nodes usually represent the current living species or other entities, such as protein sequences. The internal nodes represent the ancestors of the entities in the leaf nodes. Typically, those ancestors are not directly observable as they depict the hypothetical organisms or sequences. The lengths of edges correspond to the evolutionary distance between the nodes. Such distance usually represent the number of changes in the organisms or sequences. It is often interpreted as time estimates. However, the correspondence between the number of changes in a sequence and the elapsed time is not straightforward [90].

A phylogenetic tree can be either rooted or unrooted.

- **Rooted tree**: the root of the tree represents the most recent common ancestor of all nodes in the tree.
- **Unrooted tree**: the absence of a root can be caused by the fact that some algorithms are not able to produce rooted trees. An unrooted tree can be easily created from the rooted tree by omitting the root. On the other hand, creating a rooted tree from an unrooted tree requires special techniques, described in the section 6.3.

6.2 Methods for the phylogenetic tree construction

The input for the construction of the phylogenetic tree is a set of biological sequences. In the case when the leaf nodes represent species, the input sequences are usually highly conserved representative genes, typically rRNA [112]. In this section, three approaches for the construction of phylogenetic trees will be presented: distance-based methods, maximum parsimony and maximum likelihood methods.

6.2.1 Distance-based methods

Distance-based methods work with pairwise distances between the input sequences. These methods require an MSA as an input because the distances are computed with the use of the alignment, for example, as a number of mismatches between the sequences. The goal of these methods is to construct a tree in which the node distances are equal to the editing distances between the sequences. However, this is possible only when the distance matrix is additive. Otherwise, the exact solution does not exist and approximation techniques have to be used. The two most important distance-based methods are UPGMA and neighbour joining.

UPGMA

Unweighted pair group method with arithmetic mean (UPGMA) [103] is a simple clustering method, which can be also used for the construction of a phylogenetic tree. At the beginning, each sequence is assigned to its own cluster. The algorithm then works in iterations:

- Two nearest clusters are chosen, based on the distance matrix.
- These two clusters are merged into one bigger cluster. The distances for the new cluster in the distance matrix are computed as a proportional average of the distances to the old clusters:

  $$d_{(A \cup B),C} = \frac{|A|d_{A,C} + |B|d_{B,C}}{|A| + |B|}$$

  , where $d_{X,Y}$ denotes the distance between clusters X and Y, and $|X|$ denotes the cardinality of the cluster X.
- The merging of clusters is marked to the constructed tree.
- These steps are repeated until all sequences are merged into one cluster.

UPGMA produces rooted trees, but has two main disadvantages: it assumes that the evolutionary speed is constant, and produces trees in which the distances from the root to all leaf nodes are equal (ultrameric trees). Therefore, UPGMA is often used not on its own, but as a part of more complex algorithms. For example, it is used to produce guided trees in some algorithms for MSA, such as Clustal [12] or MUSCLE [19].

Neighbor joining

Neighbor joining [89] is a method which iteratively clusters the sequences and thus produces the phylogenetic tree. The main difference from UPGMA is that in UPGMA, the goal is to find two closest clusters, while in the neighbor joining, the goal is to find clusters which are close to each other and distant from other clusters. The additional constraint, implying that the two merged clusters must be distant from other clusters, improves the quality of the resulting tree.

The following algorithm is utilized by the neighbor joining:

- An initial star-shaped phylogenetic tree with one central node is constructed.
- The value of function $Q(x, y) = D(x, y) - u(x) - u(y)$ is computed, where $D(x, y)$ is the value from the distance matrix, and $u(x) = \frac{1}{N-2} \sum_{i \in I} D(x, i)$ represents the distance of node X from all other nodes. I is the set of all nodes and N is the number of nodes. Two nodes A, B with $min[Q(a, b)]$ are merged together into the node U, which is connected to the central node of the tree.
- The lengths of the newly created edges are $L(a, u) = 0.5 * (D(a, b) + u(a) - u(b))$ and $L(b, u) = 0.5 * (D(a, b) + u(b) - u(a))$, respectively.
- The distances of the new node from other nodes are $D(u, i) = 0.5 * (D(a, i) + D(b, i) - D(a, b))$
- The merging steps are repeated until the lengths of all edges in the tree are resolved.

The neighbor joining solves the disadvantages of UPGMA. It does not produce ultrameric trees and there is no assumption of the constant evolutionary speed. Neighbor joining constructs unrooted trees.

6.2.2 Maximum parsimony

The main idea of the maximum parsimony methods is that the tree is constructed directly from the multiple sequence alignment. The alignment contains more information than the distance matrix. Therefore, the reason to omit the distance matrices from the process is to avoid an information loss between the alignment and the distance matrix. The goal of the maximum parsimony methods is to find a tree which describes the data with the smallest number of edits (mutations). In the simplest case, every edit has the same penalty. However, in real applications, a weighted score is used in order to reflect different probabilities of various mutations.

Two tasks occur in the maximum parsimony methods: small and large parsimony problem.

- **Small parsimony problem** describes the task of assigning values to the inner nodes of the tree. In this case, the shape of the tree is assumed to be known. Solving the small parsimony problem requires the inference of ancestral sequences. Two algorithms for the small parsimony problem are Fitch and Sankoff algorithms.
- **Large parsimony problem** is a task of finding the shape of the tree for a given set of sequences. This problem is NP-hard and can not be practically solved without the use of heuristic methods, such as nearest-neighbour interchange.

Fitch algorithm

Fitch algorithm [29] solves the small parsimony problem by using recursion. In the first step, the algorithm goes from the leaf nodes to the root. For each node and position, the set of admissible amino acids is constructed as an intersection of admissible sets of its child nodes. If the intersection is empty, union is used instead. This way, the admissible sets are constructed for all nodes. In the second step, the sequences for the nodes are generated with the use of the admissible sets. An arbitrary amino acid from the admissible set is selected for the root. For all other nodes, the same amino acid as in the parent node is selected, if it is present in the admissible set. Otherwise, an arbitrary amino acid from the admissible set is selected.

Sankoff algorithm

The problem of Fitch algorithm is that it does not consider different probabilities of various mutations. This drawback is resolved by Sankoff algorithm [91], which solves the weighted small parsimony problem. The main idea of Sankoff algorithm is very similar to Fitch algorithm. In the first step, the tree is traversed from leaves to the root. For each position, the score for each amino acid is computed with the help of the scoring matrix and the values from the child nodes. In the second step, values for each node from the root to the leaves are selected with respect to the scores from the first step.

Nearest neighbor interchange

The large parsimony problem is a difficult task and has to be solved using a heuristic. One of the most prominent heuristics is the nearest neighbor interchange [95]. For every edge of the tree, it is possible to create an unrooted tree with four subtrees. Those subtrees can be connected in three different ways. Three resulting trees are called neighbors. All possible trees can be connected into one high-level graph, in which the nodes represent possible tree shapes and the edges connect neighbors described earlier. The goal is then to find the node representing the best possible tree shape. For this purpose, the graph is traversed in the following way:

- The algorithm starts in a random node.
- The parsimony score of all neighbors of the selected node is computed.
- The neighbor with the best parsimony score is selected for the next iteration of the algorithm.

Nearest neighbor interchange is a greedy algorithm. The main problem of the algorithm is the fact that it does not guarantee that the best possible tree shape is found. Most of the time, it gets stuck in a local minimum, so it has to be restarted repeatedly. Other options include a tree pruning and regrafting or a bisection and reconnection method.

6.2.3 Maximum likelihood

Maximum parsimony methods have one important issue: they do not consider different lengths of the edges. Instead, their goal is to create a tree with the lowest number of edits. Maximum likelihood uses a different strategy. The criterion is the probability that the input data come from the given phylogenetic tree. This probability is called likelihood.

Maximum likelihood methods are more precise as they incorporate substitution models, such as Jukes-Cantor model [43] or Kimura model [51]. Substitution models are Markov processes which describe the evolutionary changes. However, the use of substitution models also means that the method is computationally more expensive.

Maximum likelihood deals with three subproblems: tiny, small and large likelihood problem [42].

Tiny likelihood problem

The easiest subproblem of the maximum likelihood method is a task of evaluating the inner nodes of the tree and computing the overall likelihood score. In this case, the shape of the tree and the lengths of the edges are assumed to be known.

The algorithm for solving the tiny likelihood problem is called Felsenstein algorithm [25]. In the first step, the tree is traversed from leaves to the root and likelihood values for all possible characters in all nodes are computed. As a result, each node has a vector of likelihood values, where the length of the vector is the number of possible characters (4 for nucleotides, 20 for proteins).

- **Leaf node**: the likelihood is 1 for the character which matches the character in the node, 0 for other characters

- **Inner node**: the likelihood of the k-th character in the vector of node S_k with child nodes S_i and S_j is given by the following equation:

$$L_{S_k}(k) = \Big[\sum_{S_i} P_{S_k S_i}(t_i) \cdot L_{S_i}(i)\Big] \cdot \Big[\sum_{S_j} P_{S_k S_j}(t_j) \cdot L_{S_j}(j)\Big]$$

  , where $P_{S_k S_i}(t_i)$ is computed from the substitution model. It is a probability of a transition from the k-th character of the parent node to the i-th character of the child node, with the length of the edge t_i. L_{S_i} is a likelihood value for the i-th character of the child node.

- **Likelihood of the tree** is given by

$$L = \sum_{S_0} \pi_{S_0} \cdot L_{S_0}(0)$$

  , where π_{s_0} is the probability that the given character was selected at the beginning, and $L_{S_0}(0)$ represents the likelihood values in the root of the tree.

In the second step, the tree is traversed from the root to the leaves. Maximum of the likelihood is computed and the corresponding character is selected.

Small likelihood problem

Small likelihood problem describes a situation in which the shape of the tree is known, but the evaluation of the inner nodes and the optimal lengths of the edges are unknown. The task is more complex than the tiny likelihood problem due to the unknown edge lengths. The problem can be solved with the use of the algorithm presented by Felsenstein [25]. The algorithm uses a hill climbing technique, which iteratively improves the initial values. It is not guaranteed that the global maximum is found, but in the practical application, the results are usually sufficient.

Large likelihood problem

The most difficult task is the search for the optimal shape of the tree, together with the edge lengths and the evaluations of the inner nodes. The problem is NP-hard. A heuristic algorithm was presented by Felsenstein [25]. The edges are iteratively added to the tree and the lengths of the edges are recomputed. After a few nodes are added, the tree is reorganized in order to escape from local minimum.

6.3 Rooting

The methods described in the previous section differ in one important aspect. Some of them produce rooted phylogenetic trees while the others produce unrooted trees. Most of the practical applications require rooted trees [52] as the unrooted trees show only the relations between the species or sequences, but do not reveal the direction of the evolution. In the unrooted trees, it is impossible to find a common ancestor of a group of nodes.

One option is to use a tree construction algorithm which inherently produces rooted trees. However, most of the algorithms do not belong to this group. The exception is UPGMA, but its drawbacks outweigh the benefit of the inherent rooting. A better idea is to use a method which produces unrooted trees, such as maximum likelihood, and compute the root subsequently. Three methods for the tree rooting are presented in this section.

6.3.1 Outgroup rooting

Outgroup rooting [61] is a method which requires a biological knowledge of the input data. The main idea is that one distant sequence is added to the set of input sequences. This sequence, so-called outgroup, is known to be dissimilar to all other sequences. As a result, the root can be placed on the edge between the outgroup and the rest of the tree. The problem of the outgroup rooting is that it requires a priori knowledge. In most of the cases, this knowledge has to be provided by an expert as it is very difficult to automatize the outgroup selection.

6.3.2 Midpoint rooting

The advantage of the midpoint rooting [39] is that it does not require any expert knowledge as it selects the root solely based on the lengths of the edges. In the first step, the distances between all pairs of the leave nodes are computed and the longest path is found. The root is then placed in the middle of this path.

Midpoint rooting can be easily used in automatized pipelines. It can be also used when there is a lack of the expert knowledge for the outgroup rooting. It is often used for viral datasets [52]. However, the problem is that the midpoint rooting expects a constant rate of the evolution. When the tree is not balanced and the evolutionary rates vary, the algorithm produces wrong results.

6.3.3 Minimal ancestor deviation

The recent technique for the tree rooting is called minimal ancestor deviation [107]. When using midpoint rooting, the distances of the leaves from the root vary (unless the data respects the strict molecular clock, which is rarely the case). In the ideal situation, the root should be in the middle of the path between each pair of leaf nodes. The deviation from this ideal case can be computed by examining all pairs of leaves. Such value, so-called ancestor deviation, is a basis of the minimal ancestor deviation method. In this method, the root is placed on each branch of the tree and the ancestor deviation is computed for all such positions. The position with the lowest deviation is selected as a root of the tree.

This method is superior to the midpoint rooting because it does not require the assumption of a strict molecular clock.

6.4 Ancestral sequence reconstruction

The reconstruction of ancestral sequences is a powerful approach for designing proteins with interesting properties. For example, ancestral proteins often have higher thermal stability. These ancestors are inherently produced by some of the methods for the construction of phylogenetic trees. In the maximum parsimony method, the search for ancestors is a part of the small parsimony problem. Similarly, in the maximum likelihood method, the corresponding task is the small likelihood problem. The third possible approach is Bayesian inference [93], which is similar to maximum likelihood, but incorporates more sources of uncertainty, such as model parameters.

However, none of these methods is flawless. Maximum parsimony is the most obsolete of those three. It is computationally effective but also too simplistic. The smallest possible number of edits is a metric which is not ideal considering the underlying biological principles. Maximum likelihood partly resolves this issue, but it is very computationally demanding. Furthermore, it can realistically explore only a small fraction of all possible phylogenetic trees due to the enormous size of the state space. Bayesian inference takes the most sources of uncertainty into account, but can sometimes produce biologically irrelevant ancestors [22].

Another challenge for the reconstruction of ancestral sequences is the reconstruction of gaps. They can not be treated in the same way as the standard characters [74]. Gap reconstruction often requires manual work. There are a few algorithms for automated gap reconstruction, but their application has been limited so far [74].

Last but not least, when using an ancestral sequence reconstruction in order to find stable proteins, the crucial problem is the selection of input sequences for the phylogenetic tree construction. All in all, the process is complicated and contains many obstacles. This raises the need for the automated reconstruction workflows. One of them, FireProt$^{\mathrm{ASR}}$, is presented in the next chapter.

Chapter 7

Implementation

Ancestral sequence reconstruction is an useful method in protein engineering. However, the process requires a lot of expert knowledge and laborious manual interventions. In this chapter, a tool for fully automated ancestral reconstruction called FireProt$^{\mathrm{ASR}}$ [71] is presented. In the first part of this chapter, a workflow of FireProt$^{\mathrm{ASR}}$ is described. Then, the contributions of this thesis are presented, together with the technical details of the implementation.

7.1 FireProt$^{\mathrm{ASR}}$ workflow

FireProt$^{\mathrm{ASR}}$ is a tool developed in Loschmidt Laboratories[1] in Brno. It connects multiple tools into one fully automated workflow for ancestral sequence reconstruction. The main advantage of FireProt$^{\mathrm{ASR}}$ is the fact that it removes the manual work from the process. It is able to reconstruct ancestors even from a single sequence, while other methods usually require an MSA as an input. FireProt$^{\mathrm{ASR}}$ has a web interface and is freely available to use[2].

The workflow of FireProt$^{\mathrm{ASR}}$ is shown in Figure 7.1 and contains multiple underlying tools. The workflow consists of two main parts described in the next sections.

7.1.1 Phase 1: collection of the initial sequences

In the first phase, an initial set of homologous sequences is created. The input of the FireProt$^{\mathrm{ASR}}$ workflow is only one query sequence. The homologous sequences are selected in an automated way based on this query. Such process is crucial for the quality of the results and conventionally must be done manually with a lot of expertise.

An intuitive way for selecting homologs would be to perform a BLAST search against a database of protein sequences and choose sequences similar to the query. However, there is no guarantee that these sequences are biologically relevant to the query. Therefore, a more complex strategy has to be utilized. At the beginning, catalytic residues of the query sequence are searched using SwissProt [10] and Catalytic Site Atlas [86]. The user can adjust the selection, or continue with the automatically selected residues. Afterwards, a set of homologs is collected using EnzymeMiner [40], a tool for protein searching developed in Loschmidt Laboratories. EnzymeMiner first searches homologs of the query using PSI-BLAST, and then filters out sequences without the selected catalytic residues. Therefore,

[1]Loschmidt Laboratories: `https://loschmidt.chemi.muni.cz`

[2]FireProt$^{\mathrm{ASR}}$: `https://loschmidt.chemi.muni.cz/fireprotasr`

the biological relevance of the homologs is ensured. Alternatively, if no catalytic residues are specified, a simple BLAST search is performed instead of the use of EnzymeMiner.

The previous step generates a huge number of homologs. Therefore, this initial set must be reduced. Firstly, the sequences with lengths 20 % higher or lower than that of the query sequence are removed. Then, only sequences in a certain range of similarity to the query are kept. The user can set the threshold values of the sequence similarity. By default, the upper and lower similarity limits are set to 90 % and 30 %, respectively. These steps assure that there are no distant homologs, while the set is also unbiased towards the query.

Consequently, the sequences are clustered using USEARCH [20] and one sequence is selected from each cluster. The clustering threshold is 90 % by default and can be changed by the user. The cluster filtering reduces the number of sequences radically while maintaining the diversity of the set.

However, the number of sequences can still be too high at this point. In order to further reduce the size of the set, an initial alignment is constructed using MAFFT [47] and a phylogenetic tree is constructed from the alignment with the neighbor joining method. Treemmer [67] is then employed on the resulting tree. Treemmer is a tool which takes a phylogenetic tree as an input and iteratively prunes the leaves until the selected number of nodes remains. The nodes to be pruned are selected in a way that the diversity of the tree is preserved.

The result of the first phase is a set of 150 sequences. The user can choose to automatically submit these sequences into the second phase. Alternatively, it is possible to manually inspect the alignment and phylogenetic tree and exclude certain sequences or subtrees from the following calculations.

7.1.2 Phase 2: ancestral sequence reconstruction

In the second phase, ancestral sequences are reconstructed from a set of sequences. This set of sequences is usually collected based on one query sequence in the first phase, as described in the previous section. However, the user can also completely skip the first phase and provide his own set of sequences or even a multiple sequence alignment. Either way, the rest of the second phase is the same.

At the beginning, an alignment is constructed using ClustalΩ[99]. IQ-TREE package [77] is used to choose the most suitable evolutionary matrix. The alignment and the selected evolutionary matrix are then forwarded into RAxML [104] and the phylogenetic tree is constructed. RAxML is a tool which uses the maximum likelihood approach. It is not as fast as the neighbor joining used in the first phase, but produce more accurate trees.

RAxML constructs unrooted trees, however, for the following process a rooted tree is required. One of the options for rooting would be outgroup rooting. However, it is complicated to perform outgroup rooting in an automated way. For prokaryotic sequences, it may not be even possible because of the horizontal gene transfers. Therefore, the minimal ancestor deviation is used instead.

In the next step, ancestral sequences are reconstructed using Lazarus [36]. This tool uses PAML [113] internally – it runs and aggregates multiple PAML jobs in order to reconstruct the ancestors. After that, the ancestral gaps are reconstructed using a localized weighted back-to-consensus approach. The algorithm selects positions which are assigned as gaps. The rest of the positions are assigned with values computed by Lazarus.

Next, the 3D structure of the query sequence is modeled. This step is not an intrinsic part of the ancestral sequence reconstruction. It is used only to display the protein to the

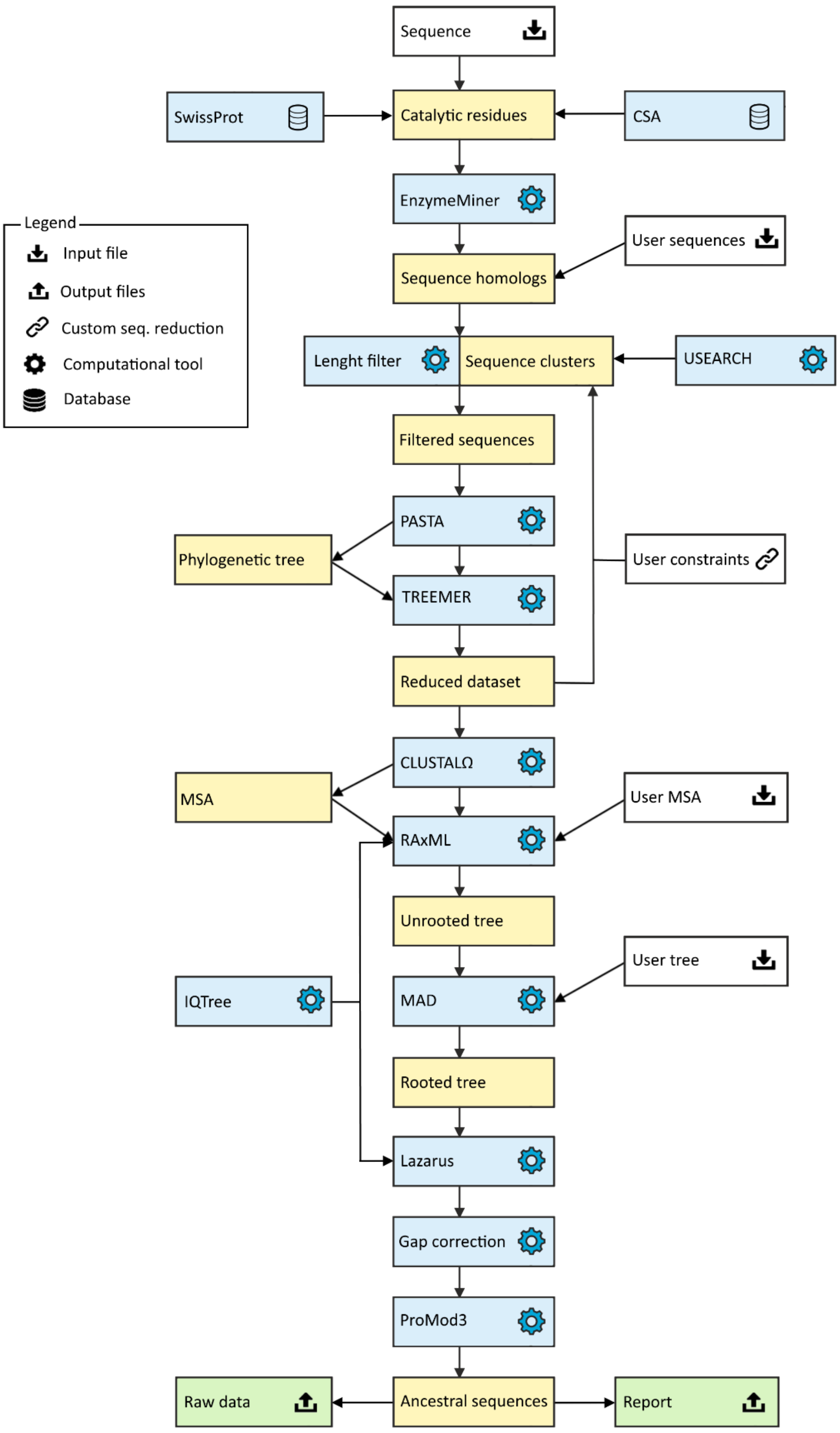


Figure 7.1: A workflow of FireProt^{ASR}. Adapted from Musil et al.[71].

user and show the differences between the query and the selected ancestor. The modelling is done using ProMod3 [9]. First, a sequence similar to the query is searched in the PDB database using BLASTp. Then, the structure from PDB is used in ProMod3 to model the structure of the query sequence.

As a last step, successors are computed. This step was added within this thesis and is described in detail in section 7.4.

Once the computations are finished, the result is visualised to the user in a web interface, which is presented in section 8.2.

7.2 FireProt^{ASR} architecture

The architecture of FireProt^{ASR} consists of three parts which exchange data: frontend, backend and the computational core. The diagram of their communication is shown in Figure 7.2. Besides the main three parts, the architecture of FireProt^{ASR} also contains a database and a high-performance computing (HPC) cluster. The goal of the database is to store the data about the computational tasks, so-called jobs, while the cluster is used to run tools which are computationally expensive.

7.2.1 Frontend

The purpose of the frontend is to collect the input data from the user and send them to the backend, which initiates the computation. After the completion of the computation, it visualizes the results. The frontend implementation details are described separately in section 7.3 as the user interface of FireProt^{ASR} is one of the main contributions of this thesis. The interface itself is described in section 8.2.

7.2.2 Backend

The backend is an intermediary between the fronted and the core. It was implemented within this thesis. The backend has a form of a HTTP server with a few endpoints and is implemented in Java with the use of the Spring Framework[3]. When the user submits a new task, so-called job, the frontend sends a request to the backend. This request contains all the input data provided by the user. The goal of the backend is to parse the data and store the representation of the job in the database. After the job is stored in the database, the computation is started by the core.

The most important task of the backend is the mediation of the result files after the computation in the core is finished. After a request from the frontend, the backend downloads the result files created by the core. Consequently, it parses these files and sends them to the frontend as a JSON response with a suitable structure. The main idea is that the frontend only displays the data, but all the parsing and data transformation is done in the backend.

7.2.3 Core

The core of FireProt^{ASR} is implemented in Java. Its purpose is to run and manage the calculations. The core checks the database in regular intervals. When it spots a new record in the database, the calculation begins with the configuration obtained from the database

[3]Spring Framework. https://spring.io/

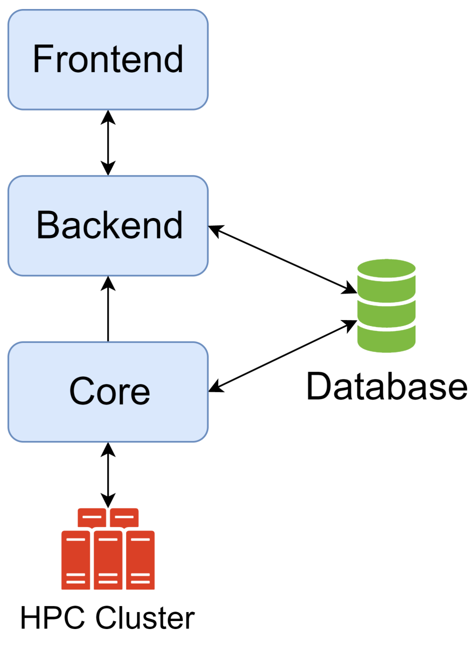


Figure 7.2: An architecture of FireProt^{ASR}

record. After the end of the calculation, the core stores the output files and updates the database record with an information about an URL where these result files are available. Specifically, the core has its own HTTP server, so the backend obtains the result files by sending requests to this server.

The main part of the core consists of so-called modules, which define various parts of the workflow presented in the section 7.1. A module typically performs the computation by calling external tools and collecting their results, however, some parts of the algorithm are computed in the core itself. FireProt^{ASR} currently has four modules:

- **Filtering** module implements the first part of the algorithm, which collects the initial sequences. It sends a request to EnzymeMiner and then collects the result. The length and similarity filters are applied and the sequences are clustered using USEARCH. The clustering is a fast task so it runs locally without the usage of the HPC cluster.
- **Reduction** is used to reduce the number of sequences. It constructs a phylogenetic tree, prunes it and constructs an MSA from the tree. All those subtasks are computationally expensive so they are performed on the HPC cluster.
- **Reconstruction** module implements the ancestral sequence reconstruction. It comprises of the tools for the construction of the phylogenetic tree, rooting of the said tree, computing the ancestral sequences and modeling the 3D structure. All these tools run on the cluster, except a minimal ancestral deviation rooting, which is a fast task and can run locally.
- **Successor** module predicts the future evolution of the sequences. The implementation of the algorithm is executed locally in the core. The description of the algorithm and successors themselves is in section 7.4.

All in all, the core implements the main algorithm of FireProt^{ASR}. In the original paper [71], the core contained three modules. The fourth module was added as a part of this thesis, together with a few minor modifications in the rest of the core.

7.3 Web interface implementation

As stated earlier, the web user interface is one of the main contributions of this thesis. The frontend is implemented in TypeScript with the extensive use of React[4]. TypeScript is a superset of JavaScript and is transpiled into JavaScript. The advantage of TypeScript is a static type checking by dint of type annotations. React is an open-source frontend JavaScript library which can be used also with TypeScript. The advantage of React is that it can be used to create single-page web applications. Such applications provide more fluent experience for the user because the page is never refreshed. Instead, it is all loaded at once at the beginning, with the exception of resources that are available only after a user request – such resources are loaded on-demand as a response to the user action.

For a state management, Redux[5] library was employed. Redux manages a global state of the application. The state is implemented as an ordinary TypeScript object and is modified by so-called reducers. A reducer is a special function which has two parameters – the current state and an action. Based on the action type, and optionally also on the action value, the reducer modifies the state.

In React, the basic building blocks of the code are so-called components, which represent reusable pieces of code. In the older versions of React, class-based components were used, but a more modern functional approach was selected for FireProt^{ASR} frontend. The advantage of functional components is a more straightforward syntax and a better reusability of the code. A functional component is an ordinary TypeScript function, which returns so-called JSX. JSX is a syntax extension to JavaScript (or TypeScript). It allows mixing HTML code with JavaScript expressions in a syntactically suitable way, which leads to a better readability of the code. JSX elements are rendered into DOM elements and React is able to do this rendering effectively, so the elements are not updated when it is not necessary. With the use of JSX, it is possible to nest components and effectively build a complex page from multiple simple components.

The components which need to access the global state are wrapped in special components called containers. The purpose of containers is to pass the data between the components and Redux. It makes the state available in the underlying components, and it also provides means of modifying the state from the components. When using containers, React is able to automatically update the component when the global state changes.

For the visual part of the interface, Reactstrap[6] library is used. It offers styles from Bootstrap[7], a popular CSS framework, in a convenient form of React components.

The frontend uses three external libraries for the visualisation of biological data. Phylotree.js [98] is used for the phylogenetic tree visualization. The second one is a sequence logo visualization library with a glyph-based approach [62]. Both of these libraries are used in modified versions, developed in Loschmidt Laboratories. Additionally, for the visualization of a 3D protein structure, an external library Mol* [94] is used.

[4]React: a JavaScript library for building user interfaces. https://reactjs.org/

[5]Redux: a predictable state container for JS apps. https://redux.js.org/

[6]Reactstrap: https://reactstrap.github.io/

[7]Bootstrap: https://getbootstrap.com/

7.4 Successor analysis

The phylogenetic analysis can suggest mutations which improve the features of protein sequences. So far, the scope of this thesis has been limited to ancestral sequences. Those sequences typically have higher thermal stability, but often offer lower activity with higher substrate promiscuity. Such trade-off is often beneficial in the industrial applications. However, it would be also interesting to obtain the sequences which occur in the other evolutionary direction. Such sequences are called successors. Figure 7.3 shows the relationship between ancestors and successors. The evolution seems to increase specificity of proteins, therefore, successors are supposed to have higher activity than current days proteins. Nevertheless, the topic of successors is not as researched as the ancestral reconstruction so the amount of available information is limited.

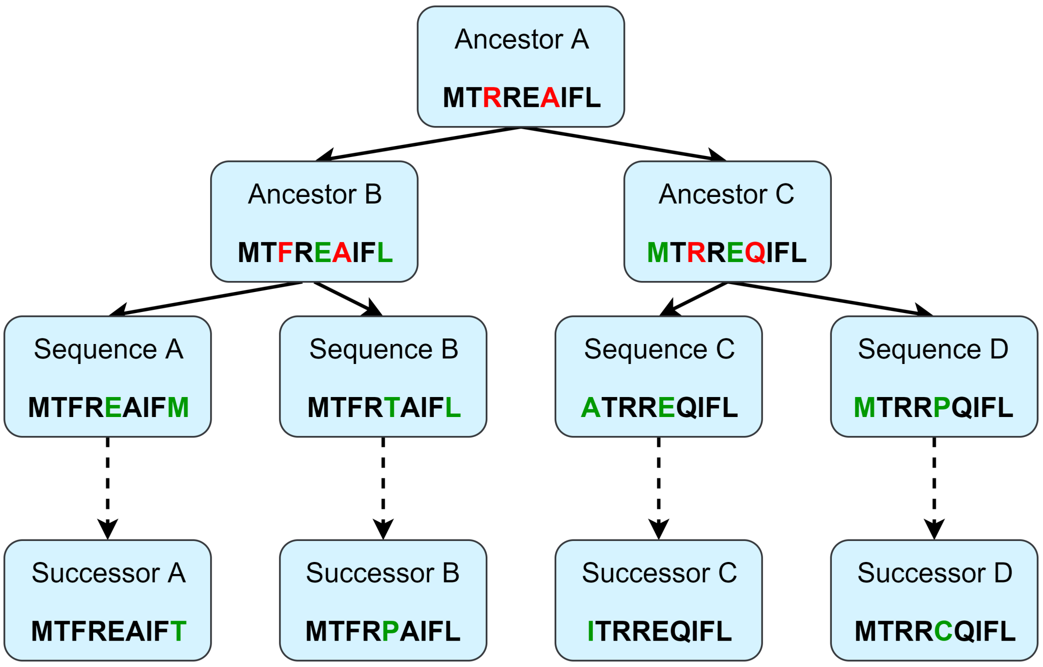


Figure 7.3: An example of a phylogenetic tree with ancestors and successors.

The successor prediction in FireProt^{ASR} runs after the ancestral reconstruction because it uses ancestral sequences from the reconstruction as an input. From the implementation point of view, the successor module contains two important algorithms, supplemented with a few auxiliary methods. Firstly, the input files must be loaded. A phylogenetic tree is parsed from a file in Newick format whereas an MSA is stored in FASTA format. While parsing a FASTA file is a trivial task, parsing Newick format is more complex and is described later in this section.

After the tree and MSA sequences are parsed, the nodes in the tree are matched with the ancestral sequences from the MSA. The tree is then traversed and successors are computed with the help of ancestors.

Newick parser

Newick [24] is a standard format used for representing tree graphs, including phylogenetic trees. It always represents a rooted tree. An unrooted tree can be converted to rooted by arbitrarily selecting the root. Newick format has many modifications, such as New Hampshire X format or Extended Newick, but the most popular is the basic Newick variant. Formally, the format defines a context-free language of all valid trees. The language is generated by the following context-free grammar:

```
Tree → Subtree ;
Subtree → Name | ( BranchSet ) Name
BranchSet → Branch | Branch , BranchSet
Branch → Subtree Length
Name → empty | string | ' string '
Length → empty | : number
```

, where | represents the alternation and other characters are literals. Newick allows a minimalistic representation of a tree shape, as well as more verbose forms with named nodes and edge lengths. For example, the tree in Figure 7.4 can be represented by many different ways, such as:

```
((,),());
((D,E),(F));
((:0.1,:0.2):0.4,(:0.3):0.5);
((D:0.1,E:0.2)B:0.4,(F:0.3)C:0.5)A;
```

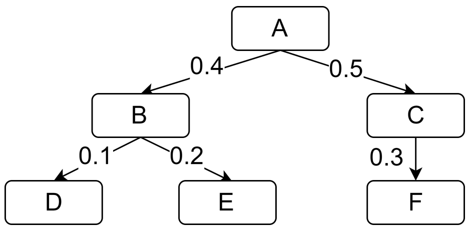


Figure 7.4: An example of a phylogenetic tree

The implementation of a Newick parser uses `StreamTokenizer` from Java standard library, which parses the input string into tokens. In this case, the tokens are mostly single characters, with an exception of a `WORD` token, which represents either the name of a node or the edge length. The stream of tokens is parsed serially. First, every token is checked by function `checkToken`, which controls whether the token is allowed with respect to the previous token. This function uses Table 7.1, which was created using the formal Newick grammar.

In the main part of the parsing algorithm, an action is performed for each token based on the token type. Although the tree is a recursive structure, the parser itself does not use a stack. It only stores the actual innermost node, which is currently parsed. Each node stores references to its parent and children, so the previously parsed nodes are accessible from the actual node. The algorithm is shown in the following pseudocode. Nevertheless, it is a simplified version: for example, checks, whether the opening and closing parentheses match, are omitted for simplicity.

```
root := new TreeNode(parent=null)
actual := root
settingEdgeLength := false
for each token
    if token is not allowed based on the table
        return Error

    switch (token.type)
        WORD
            if settingEdgeLength
                actual.setEdgeLength(token.value)
                settingEdgeLength := false
            else
                actual.setName(token.value)
        ':'
            settingEdgeLength := true
        ';'
            return root
        '('
            newChild := new TreeNode(parent=actual)
            actual.addChild(newChild)
            actual := newChild
        ')'
            actual := actual.parent
        ','
            newChild := new TreeNode(parent=actual.parent)
            actual.parent.addChild(newChild)
            actual := newChild
```

Successor prediction

Once the phylogenetic tree and the MSA are parsed, the actual prediction of successor sequences begins. As a first step, the tree is traversed from the root to the leaves and every node obtains the sequences of all its ancestors. In the second step, the ancestor sequences are used to predict the successor. This step is repeated for all leaf nodes. Each leaf is processed independently, as the prediction is based solely on the ancestor sequences in the path between the root and the given leaf. Furthermore, each position in the sequence is also processed independently. It is possible because all the sequences in the phylogenetic tree (even the ancestors) come from one MSA, so all the sequences have the same length and there is a evolutionary correspondence between the amino acids on the same positions.

The main idea is that the path from the root to the leaf represents an evolutionary history of the sequence. The changes of amino acids on each position define a trend, a direction, in which the evolution is heading. By observing this trend, it is possible to extrapolate and predict the next value, assuming that the trend continues. For example, when the hydrophobicity of amino acids was gradually increasing during the evolution, the algorithm should detect this trend and predict yet another increase in hydrophobicity for the successor amino acid.

Current token	Allowed next token				
Start of input	WORD	(			
WORD	:	;	)	,	
:	WORD				
;					
(	WORD	:	(	,	
)	WORD	:	;	)	,
,	WORD	:	(	)	,

Table 7.1: Allowed tokens in Newick parser.

For the previous idea, it is important to select a suitable representation of amino acids in order to quantify the trend and correctly extrapolate the value. In the example, a hydrophobicity was mentioned, however, it is not the only important amino acid feature.

In the end, a set of three most important amino acid features was selected as an amino acid representation. In other words, each amino acid is represented as a vector in a 3D Euclidean space. The selected features are hydrophobicity, charge and volume and their values were obtained from AAindex [49]. These values have different orders of magnitude. For example, the range for hydrophobicity is 0 – 2.65 whereas the volume values range is 90 – 228.2. It is therefore necessary to normalize the values. Min-max normalization is utilized for this task. The normalized value is computed as

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

, where x_{min} and x_{max} represent the lowest and highest value of the feature, respectively. The normalization maps the initial feature values from AAindex into the range $[0, 1]$.

The main part of the algorithm predicts one character of the successor and is schematically shown in Figure 7.5. It has two inputs: the history of amino acids on a given position on a path between the root and the given node, and the lengths of the edges on that path in the phylogenetic tree. It starts with counting the number of gaps in the value history. If a gap occurs in more than a half of ancestors, the predicted value for the successor is a gap. Otherwise, a trend of a feature value is computed. The trend $\overline{x}$ is defined as an average change of a feature value x on a unit of edge length, where x represents either hydrophobicity, charge or volume. The trend between nodes 1 and n can be computed using the individual changes of x between the consecutive nodes on a path from node 1 to node n

$$\overline{x} = \frac{\Delta_{e_1}x + \Delta_{e_2}x + \ldots + \Delta_{e_{n-1}}x}{|e_1| + |e_2| + \ldots + |e_{n-1}|}$$

, where $\Delta_{e_i}x$ denotes the change of feature x on edge e_i, and $|e_i|$ denotes the length of edge e_i. However, it is possible to simplify the expression. $\Delta_{e_i}x$ can be rewritten as a difference between the value of x for two consecutive nodes: $\Delta_{e_i}x = x_{i+1} - x_i$.

$$\overline{x} = \frac{(x_2 - x_1) + (x_3 - x_2) + \ldots + (x_n - x_{n-1})}{|e_1| + |e_2| + \ldots + |e_{n-1}|} = \frac{x_n - x_1}{|e_1| + |e_2| + \ldots + |e_{n-1}|}$$

As a result, the trend depends only on a value of x for the first and the last node, and on the sum of edge lengths between these nodes. Furthermore, a gap does not have any

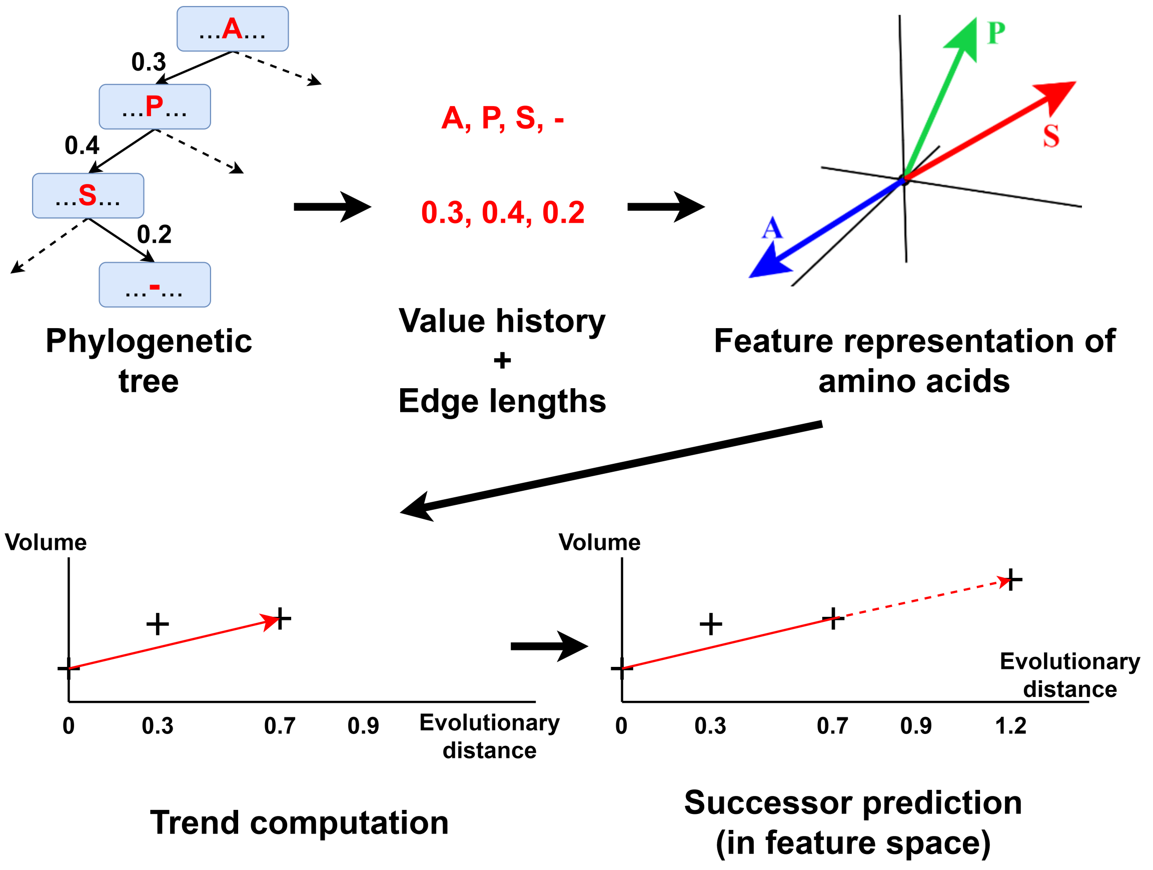


Figure 7.5: Schema of successor prediction.

representation in the 3D feature space. Therefore, the trend is computed only from the path between the first and the last non-gap character.

After the trend is computed for all three features, the successor amino acid is predicted. More precisely, a feature representation of such amino acid is computed. The successor represents a new node, which is a child of a leaf node in the current tree. The length of the new edge is computed as an average edge length on a path between the root and the leaf. This average length is supposed to represent an average evolutionary step, so the successor is placed one step further beyond the current leaf. In the simple case when the leaf does not contain a gap, the successor feature value is computed as

$$x_S = x_n + \overline{x}\left(\frac{1}{N-1}\sum_{i=2}^{N}|e_{i-1}|\right)$$

, where N is the number of nodes from the root to the leaf. The expression in brackets represent the length of the new edge. In a more general case, another expression must be added to the edge length. Such expression represents the distance between the last non-gap character and the leaf:

$$x_S = x_{lng} + \overline{x}\left(\sum_{i=lng}^{N}|e_{i-1}| + \frac{1}{N-1}\sum_{i=2}^{N}|e_{i-1}|\right)$$

, where lng is an index of the last non-gap node.

As a last step, the amino acid is selected based on the predicted feature representation, which can be viewed as a point in the feature space. The amino acid, whose feature representation is the most similar to such point, is selected. The similarity is defined in terms of a root-mean-square error across all three feature dimensions. In other words, the amino acid with the shortest Euclidean distance from the predicted point in the feature space is selected.

7.5 Latent-space search

The second suggested novel technique for FireProt$^{\mathrm{ASR}}$, besides the successor prediction, was a latent-space search. This idea is often used in machine learning. The input is compressed into a lower-dimensional representation in a so-called latent space. The mapping of the inputs is not arbitrary, because the latent representations of similar inputs should be close to each other in the latent space. The mapping is supposed to preserve the important (often hidden) features while reducing the dimensionality.

Furthermore, the latent space representation can be used to reconstruct the original input, and even to generate new data points in the original space. This is a concept of an autoencoder. Its basic schema is shown in Figure 7.6. It has two main parts – encoder and decoder. They are often implemented as neural networks.

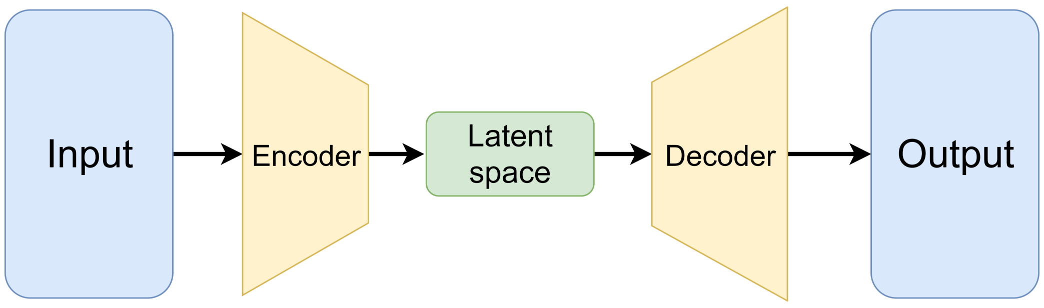


Figure 7.6: An autoencoder.

The idea of incorporating the latent-space search into FireProt$^{\mathrm{ASR}}$ was based on an assumption that using the suitable lower-dimensional representation will improve the quality of the reconstructed ancestors. However, the laboratory experiments in Loschmidt laboratories showed that using the latent space does not lead to better results and more stable ancestors. Therefore, it was decided that the method will not be implemented into FireProt$^{\mathrm{ASR}}$.

Chapter 8

Results

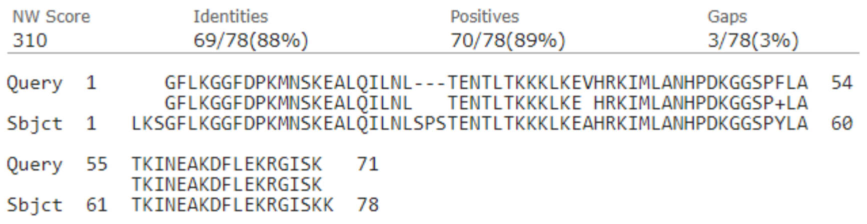
In the previous chapter, the implementation of FireProt^{ASR} was described, together with novel algorithms implemented in FireProt^{ASR}. In this chapter, the results of the implementation are described. The main focus is on two important parts: ancestral reconstruction with successors, and the new user interface.

8.1 Reconstructed sequences

The main goal of ancestral sequence reconstruction is to create proteins with higher stability. Similarly, the goal of successor prediction is to create proteins with higher activity. In an ideal case, the stability and activity of ancestors and successors would be examined in laboratory experiments. However, due to the time demands of such an experiments, the results of the laboratory measurements are still pending. The only available experimental results are from the original FireProt^{ASR} paper [71]. These experiments were made with ancestral proteins from haloalkane dehalogenase subfamily II. It was shown that the thermal stability of the inferred ancestors was 20–26°C higher than that of the wild proteins.

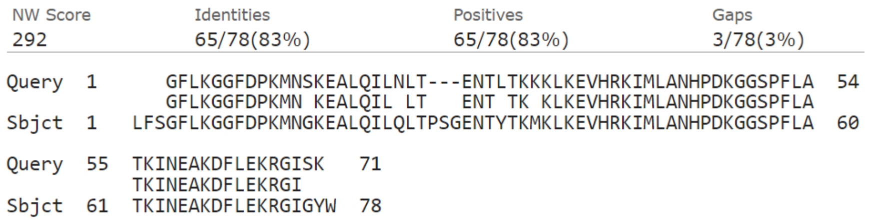
Unfortunately, it is complicated to reliably evaluate how the mutations suggested by FireProt^{ASR} contribute to the stability and activity of the protein. Therefore, this section focuses mainly on a general characterization of ancestors and successors generated by FireProt^{ASR}.

The experiments were performed on two proteins. The first is mitochondrial import inner membrane translocase subunit TIM14 (PDB code 2GUZ), which was selected because of its small size of only 71 amino acids. The second protein is gamma-glutamyl hydrolase (PDB code 1L9X), which is a representative of hydrolases, one of the most important group of enzymes.



```
NW Score         Identities         Positives          Gaps
310              69/78(88%)         70/78(89%)         3/78(3%)

Query  1      GFLKGGFDPKMNSKEALQILNL---TENTLTKKKLKEVHRKIMLANHPDKGGSPFLA  54
              GFLKGGFDPKMNSKEALQILNL   TENTLTKKKLKE HRKIMLANHPDKGGSP+LA
Sbjct  1   LKSGFLKGGFDPKMNSKEALQILNLSPSTENTLTKKKLKEAHRKIMLANHPDKGGSPYLA  60

Query  55  TKINEAKDFLEKRGISK   71
           TKINEAKDFLEKRGISK
Sbjct  61  TKINEAKDFLEKRGISKK  78
```

Figure 8.1: The pairwise alignment between the query sequence and an ancestor reconstructed using FireProt^{ASR} (denoted as `Sbjct`)

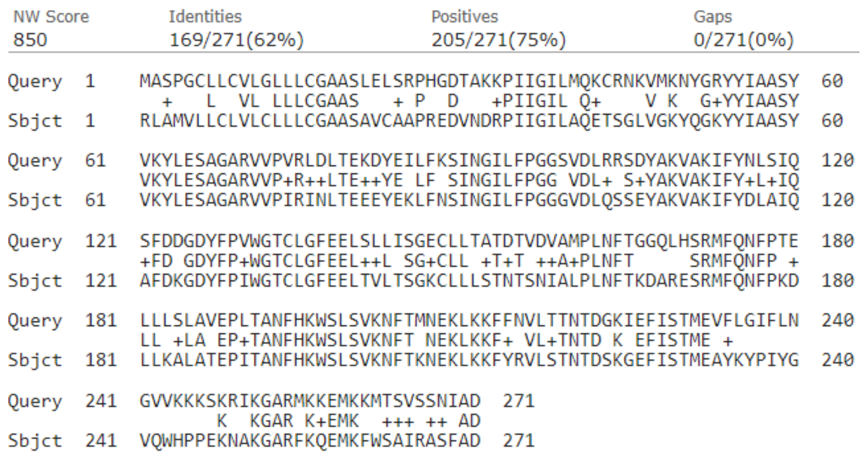


```
NW Score        Identities          Positives           Gaps
292             65/78(83%)          65/78(83%)          3/78(3%)

Query  1      GFLKGGFDPKMNSKEALQILNLT---ENTLTKKKLKEVHRKIMLANHPDKGGSPFLA  54
              GFLKGGFDPKMN KEALQIL LT   ENT TK KLKEVHRKIMLANHPDKGGSPFLA
Sbjct  1   LFSGFLKGGFDPKMNGKEALQILQLTPSGENTYTKMKLKEVHRKIMLANHPDKGGSPFLA  60

Query  55  TKINEAKDFLEKRGISK   71
           TKINEAKDFLEKRGI
Sbjct  61  TKINEAKDFLEKRGIGYW  78
```

Figure 8.2: The pairwise alignment between the query sequence and a successor reconstructed using FireProt^{ASR} (denoted as `Sbjct`)

Figures 8.1 and 8.2 show the alignment of a query sequence with the parent and successor sequences, respectively. The alignment was created using Needleman-Wunsch algorithm, the query sequence is a membrane translocase protein mentioned earlier.

It can be seen that the successor alignment has the same number of gaps as the ancestor alignment. Furthermore, it has slightly lower sequence identity. This is caused by the lengths of the edges in the phylogenetic tree – the edge between the ancestor and the query is shorter than that between the query and the successor. The conclusion is that the newly implemented successor prediction is consistent with the ancestral reconstruction in FireProt^{ASR}.

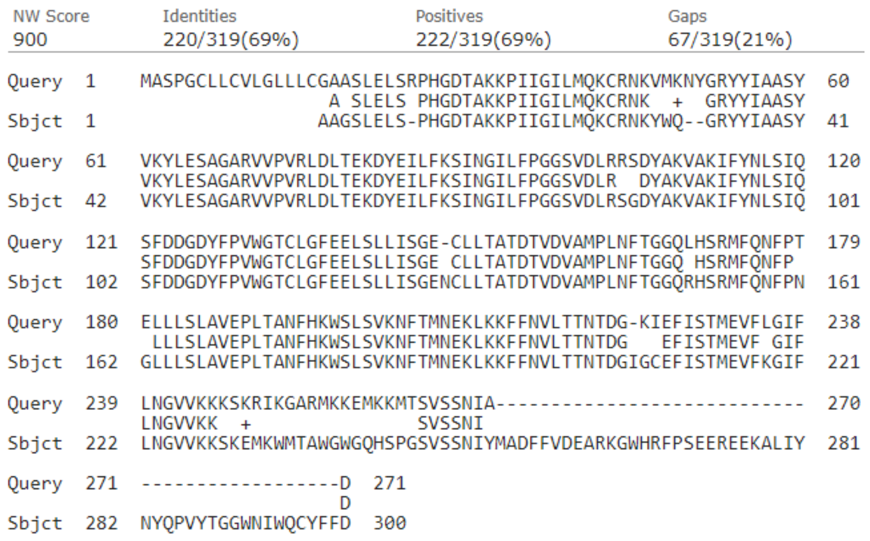


```
NW Score        Identities          Positives           Gaps
850             169/271(62%)        205/271(75%)        0/271(0%)

Query  1    MASPGCLLCVLGLLLCGAASLELSRPHGDTAKKPIIGILMQKCRNKVMKNYGRYYIAASY  60
              +   L  VL LLLCGAAS   + P  D   +PIIGIL Q+    V K  G+YYIAASY
Sbjct  1    RLAMVLLCLVLCLLLCGAASAVCAAPREDVNDRPIIGILAQETSGLVGKYQGKYYIAASY  60

Query  61   VKYLESAGARVVPVRLDLTEKDYEILFKSINGILFPGGSVDLRRSDYAKVAKIFYNLSIQ  120
            VKYLESAGARVVP+R++LTE++YE LF SINGILFPGG VDL+ S+YAKVAKIFY+L+IQ
Sbjct  61   VKYLESAGARVVPIRINLTEEEYEKLFNSINGILFPGGGVDLQSSEYAKVAKIFYDLAIQ  120

Query  121  SFDDGDYFPVWGTCLGFEELSLLISGECLLTATDTVDVAMPLNFTGGQLHSRMFQNFPTE  180
            +FD GDYFP+WGTCLGFEEL++L SG+CLL +T+T ++A+PLNFT     SRMFQNFP +
Sbjct  121  AFDKGDYFPIWGTCLGFEELTVLTSGKCLLLSTNTSNIALPLNFTKDARESRMFQNFPKD  180

Query  181  LLLSLAVEPLTANFHKWSLSVKNFTMNEKLKKFFNVLTTNTDGKIEFISTMEVFLGIFLN  240
            LL +LA EP+TANFHKWSLSVKNFT NEKLKKF+ VL+TNTD K EFISTME +
Sbjct  181  LLKALATEPITANFHKWSLSVKNFTKNEKLKKFYRVLSTNTDSKGEFISTMEAYKYPIYG  240

Query  241  GVVKKKSKRIKGARMKKEMKKMTSVSSNIAD  271
                   K  KGAR K+EMK  +++ ++ AD
Sbjct  241  VQWHPPEKNAKGARFKQEMKFWSAIRASFAD  271
```

Figure 8.3: The pairwise alignment between the query sequence and an ancestor reconstructed using FireProt^{ASR} (denoted as `Sbjct`)

A similar pair of alignments is presented also for the gamma-glutamyl hydrolase, as seen in Figures 8.3 and 8.4. In this case, the successor alignment contains a significant number of gaps, while the ancestor alignment does not have any gaps. The sequence identity is similar in both cases.

To conclude, the successor prediction is able to suggest mutations in a rate similar to the rate of mutations in ancestors. Assuming that the ancestral reconstruction in FireProt^{ASR} works correctly, which was proved by the laboratory experiments, the new successor analysis also produces meaningful results. Furthermore, the algorithm inherently preserves conserved residues because it predicts mutations only when the amino acid on a certain position was not conserved. It suggests that the functional properties of the protein are also preserved.



```
NW Score        Identities          Positives           Gaps
900             220/319(69%)        222/319(69%)        67/319(21%)

Query  1    MASPGCLLCVLGLLLCGAASLELSRPHGDTAKKPIIGILMQKCRNKVMKNYGRYYIAASY  60
                             A SLELS PHGDTAKKPIIGILMQKCRNK  +  GRYYIAASY
Sbjct  1                    AAGSLELS-PHGDTAKKPIIGILMQKCRNKYWQ--GRYYIAASY  41

Query  61   VKYLESAGARVVPVRLDLTEKDYEILFKSINGILFPGGSVDLRRSDYAKVAKIFYNLSIQ  120
            VKYLESAGARVVPVRLDLTEKDYEILFKSINGILFPGGSVDLR  DYAKVAKIFYNLSIQ
Sbjct  42   VKYLESAGARVVPVRLDLTEKDYEILFKSINGILFPGGSVDLRSGDYAKVAKIFYNLSIQ  101

Query  121  SFDDGDYFPVWGTCLGFEELSLLISGE-CLLTATDTVDVAMPLNFTGGQLHSRMFQNFPT  179
            SFDDGDYFPVWGTCLGFEELSLLISGE CLLTATDTVDVAMPLNFTGGQ HSRMFQNFP
Sbjct  102  SFDDGDYFPVWGTCLGFEELSLLISGENCLLTATDTVDVAMPLNFTGGQRHSRMFQNFPN  161

Query  180  ELLLSLAVEPLTANFHKWSLSVKNFTMNEKLKKFFNVLTTNTDG-KIEFISTMEVFLGIF  238
             LLLSLAVEPLTANFHKWSLSVKNFTMNEKLKKFFNVLTTNTDG   EFISTMEVF GIF
Sbjct  162  GLLLSLAVEPLTANFHKWSLSVKNFTMNEKLKKFFNVLTTNTDGIGCEFISTMEVFKGIF  221

Query  239  LNGVVKKKSKRIKGARMKKEMKKMTSVSSNIA---------------------------  270
            LNGVVKK  +                SVSSNI
Sbjct  222  LNGVVKKKSKEMKWMTAWGWGQHSPGSVSSNIYMADFFVDEARKGWHRFPSEEREEKALIY  281

Query  271  ------------------D  271
                              D
Sbjct  282  NYQPVYTGGWNIWQCYFFD  300
```

Figure 8.4: The pairwise alignment between the query sequence and a successor reconstructed using FireProt^{ASR} (denoted as `Sbjct`)

8.2 User interface

The user interface has a form of a single-page web application. However, the application contains multiple routes within the single-page architecture. It means that from the user point of view, it looks that there are various distinct pages. However, the benefits of the single-page application still apply because all the resources are loaded from the server at once at the beginning. Therefore, when the user moves from one page to another, the application is not refreshed and the experience is fluent. In this chapter, the term *page* denotes one route of the single page application, such as input page or result page.

The first important page of the application is the **input page**, shown in Figure 8.5. There, the user creates a new ancestral reconstruction task (so-called job). The user can choose one of the three input types. The most simple is a sequence input. In that case, the user provides only one sequence, which means that the whole FireProt^{ASR} algorithm will run, including the collection of homologs and the sequence filtering. The other two input options allow the user to skip these steps by providing their own set of sequences or MSA. Only the ancestral reconstruction step is performed in such case. The input page also offers an option to set a job title or email.

The input itself is designed with emphasis on the intuitiveness. When the user types the sequence, the input is dynamically checked. When the sequence does not meet the required conditions, e.g. it contains a forbidden character, it is denoted by a red border around the input box. Furthermore, the user also has an option to load the sequence from a local file.

After submitting the sequence, the user can tweak the algorithm parameters on a **calculation settings page** (Figure 8.6). The most important part is the selection of essential residues. The default residue selection is created by querying SwissProt [10] and Catalytic Site Atlas [86], however, the user can modify this selection or even add his own residues.

Apart from the residue selection, the user also has an option to choose between the one-step and two-step calculation. Two-step calculation means that the user can modify the set of sequences collected in the first part of the algorithm.

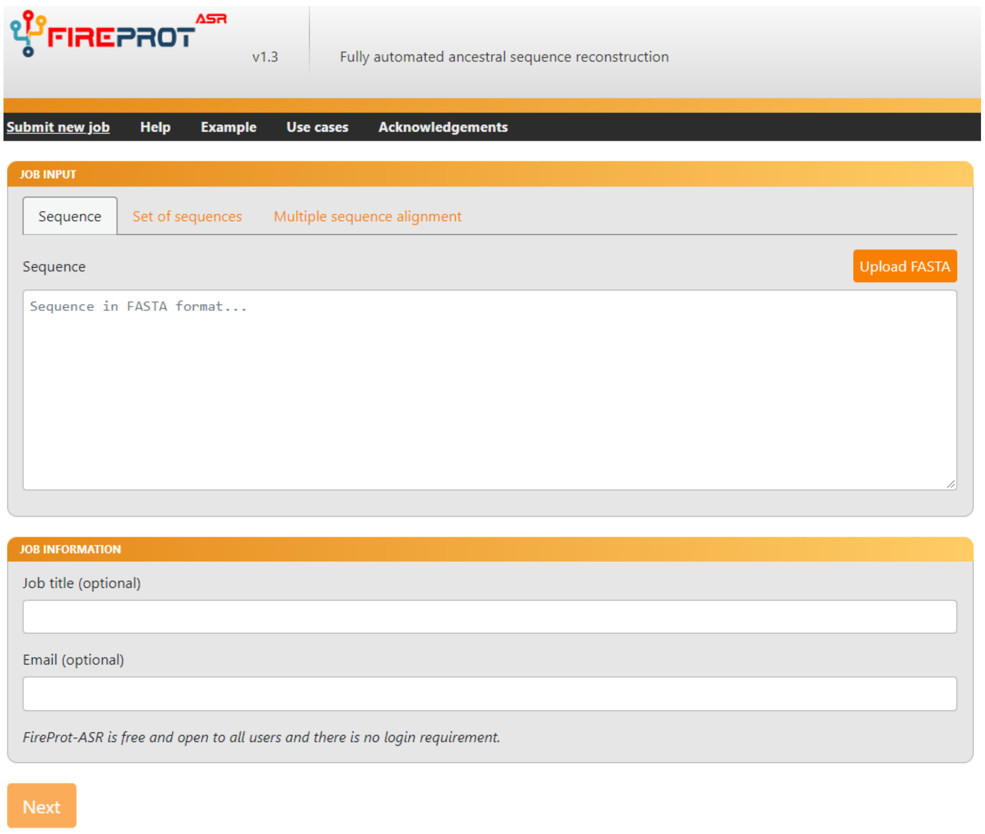

FIREPROT ASR
v1.3
Fully automated ancestral sequence reconstruction
Submit new job
Help
Example
Use cases
Acknowledgements
JOB INPUT
Sequence
Set of sequences
Multiple sequence alignment
Sequence
Upload FASTA
Sequence in FASTA format...
JOB INFORMATION
Job title (optional)
Email (optional)
FireProt-ASR is free and open to all users and there is no login requirement.
Next

Figure 8.5: The input page of FireProt$^{\text{ASR}}$.

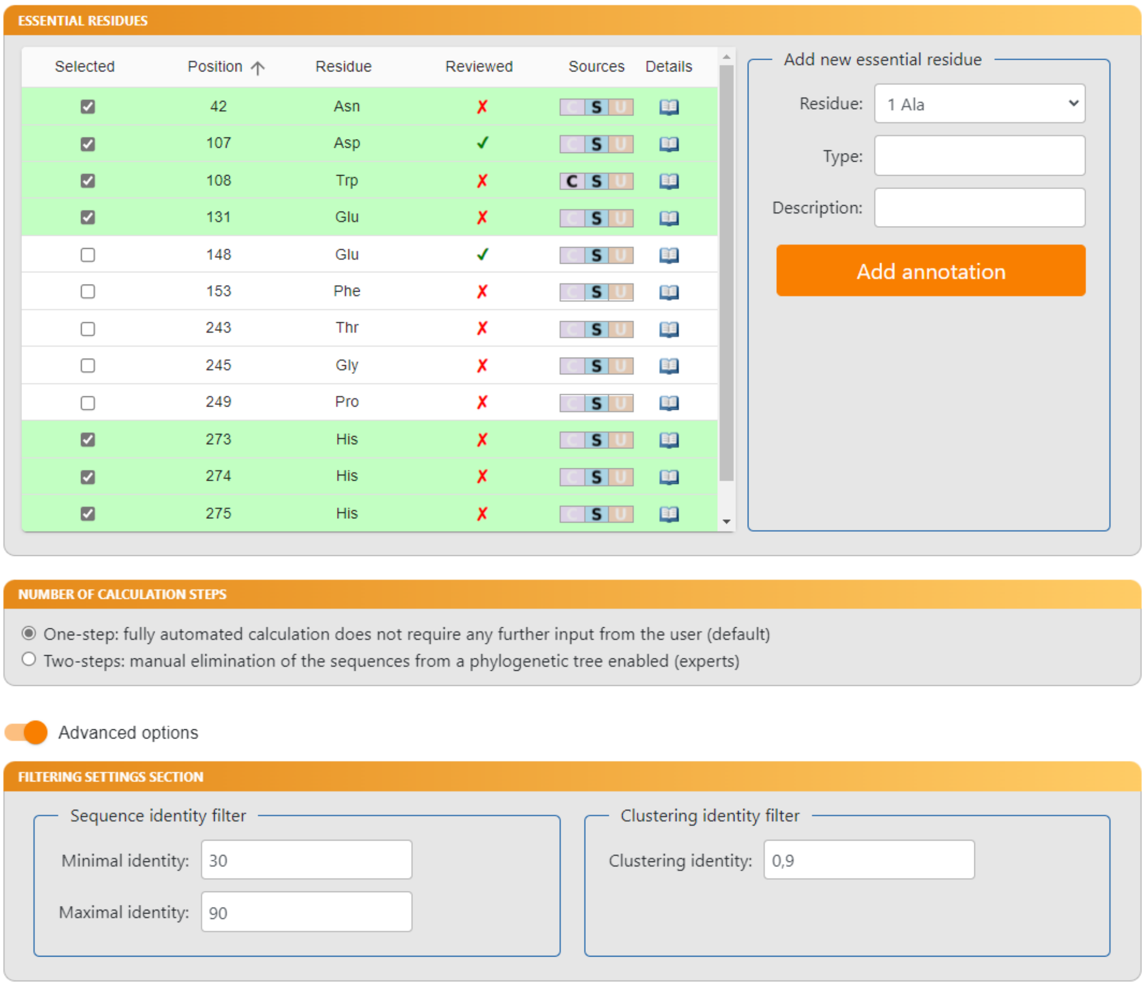
Furthermore, it is possible to adjust the parameters of the algorithm, such as minimal and maximal sequence identities, which are used in the sequence collection step.

After the user sets the calculation settings, the job is submitted. An unique ID is created for the job. The user can save the page as a bookmark, or just write down the ID so that he can reach the page later when the results are ready. When the user opens a page representing a particular job, the job information is loaded from the database. When the job is finished, the result is visualised. In a case of a two-step calculation, a **sequence filtering page** is displayed. The sequences collected in the first step of the algorithm are presented there in a form of an MSA and also in a phylogenetic tree. An example of an MSA is shown in Figure 8.7. The user can choose the color scheme – the available schemes are Clustal, Taylor and Zappo.

On the sequence filtering page, the user can eliminate certain sequences from the future calculation. This can be done either from the MSA visualisation, or from the phylogenetic tree visualisation. In such case, it is also possible to remove whole subtree or select only a given subtree, as seen in Figure 8.8.

After the sequences are selected, the user submits them for the second step of the calculation. When the second step is finished, the result is displayed. Alternatively, if a one-step calculation was chosen, or the initial input was a set or sequences or MSA, the sequence filtering page is completely skipped. Either way, the result is visualised in the **result page**.

The result page consists of three important parts. The first is a visualisation of mutations on a 3D structure of a protein, shown in Figure 8.9. The 3D structure is modeled



ESSENTIAL RESIDUES

Selected	Position ↑	Residue	Reviewed	Sources	Details
☑	42	Asn	✗	C S U	📖
☑	107	Asp	✓	C S U	📖
☑	108	Trp	✗	C S U	📖
☑	131	Glu	✗	C S U	📖
☐	148	Glu	✓	C S U	📖
☐	153	Phe	✗	C S U	📖
☐	243	Thr	✗	C S U	📖
☐	245	Gly	✗	C S U	📖
☐	249	Pro	✗	C S U	📖
☑	273	His	✗	C S U	📖
☑	274	His	✗	C S U	📖
☑	275	His	✗	C S U	📖

Add new essential residue

Residue: 1 Ala

Type:

Description:

Add annotation

NUMBER OF CALCULATION STEPS

[✓ One-step: fully automated calculation does not require any further input from the user (default)]

Two-steps: manual elimination of the sequences from a phylogenetic tree enabled (experts)

Advanced options

FILTERING SETTINGS SECTION

Sequence identity filter

Minimal identity: 30

Maximal identity: 90

Clustering identity filter

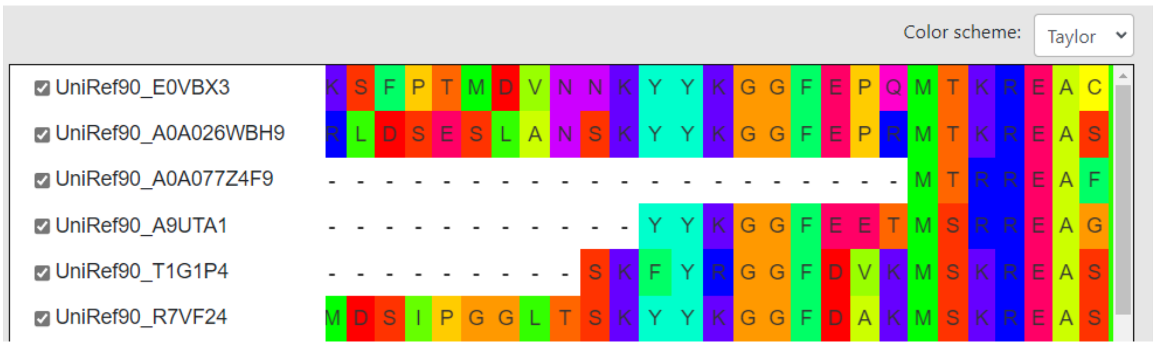
Clustering identity: 0,9

Figure 8.6: The calculation settings.

for the query protein. On the structure, the mutations between the query and a selected ancestral sequence are highlighted. The user can select from three types of mutations – substitutions, deletions or insertions. The pairwise mutations are computed on-demand using Needleman-Wunsch algorithm. The visualisation can be useful for a fast inspection, how the reconstructed ancestor differs from the query.

The second part of the result page consists of the phylogenetic tree and MSA visualisations. They don't differ very much from the visualisations on a sequence filtering page described earlier (Figures 8.7 and 8.8). The only differences are the available options in the phylogenetic tree, as seen in Figure 8.10. In this case, the user can modify the reconstructed sequence of an ancestor (i.e. node of the tree). It is also possible to reset the modified sequence back to the original values. Furthermore, the user can store the ancestral sequence of a node or display the mutations on a 3D structure.

The third part shows the stored sequences. The user can store the reconstructed sequences directly, or modify the sequences before storing. This can be done through a sequence logo, which is shown in Figure 8.11. It can be used to inspect the posterior probabilities of amino acids in the ancestor. When the user clicks on a certain position, it is possible to choose the amino acid, as seen in Figure 8.12. The stored sequences can be downloaded in a zip archive.


Color scheme: Taylor

```
☑ UniRef90_E0VBX3       K S F P T M D V N N K Y Y K G G F E P Q M T K R E A C
☑ UniRef90_A0A026WBH9   R L D S E S L A N S K Y Y K G G F E P R M T K R E A S
☑ UniRef90_A0A077Z4F9   - - - - - - - - - - - - - - - - - - - - M T R R E A F
☑ UniRef90_A9UTA1       - - - - - - - - - - - - Y Y K G G F E E T M S R R E A G
☑ UniRef90_T1G1P4       - - - - - - - - - - S K F Y R G G F D V K M S K R E A S
☑ UniRef90_R7VF24       M D S I P G G L T S K Y Y K G G F D A K M S K R E A S
```

Figure 8.7: The visualisation of an MSA.

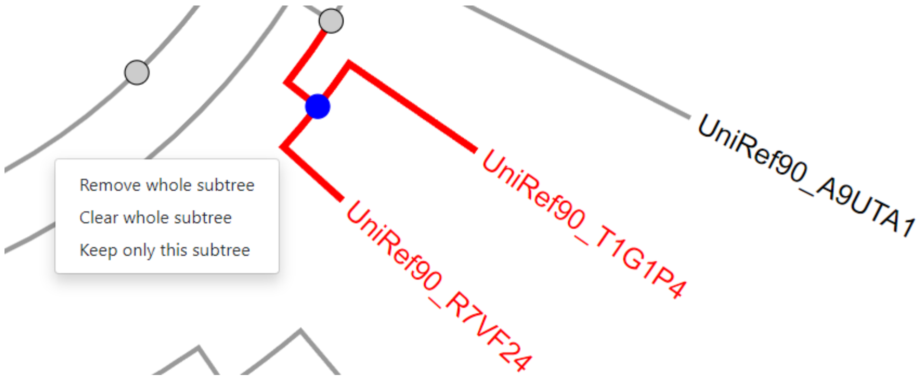


Figure 8.8: The options for a phylogenetic tree node on a sequence filtering page.

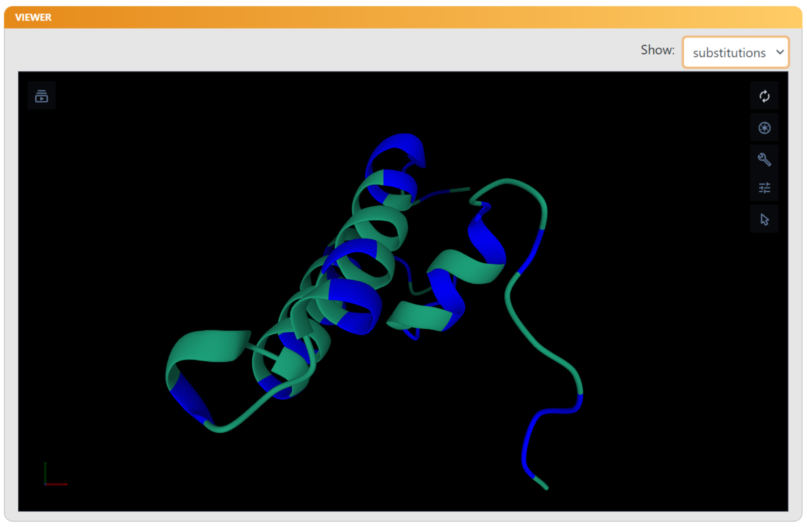


Figure 8.9: The visualisation of substitutions on a 3D protein structure.

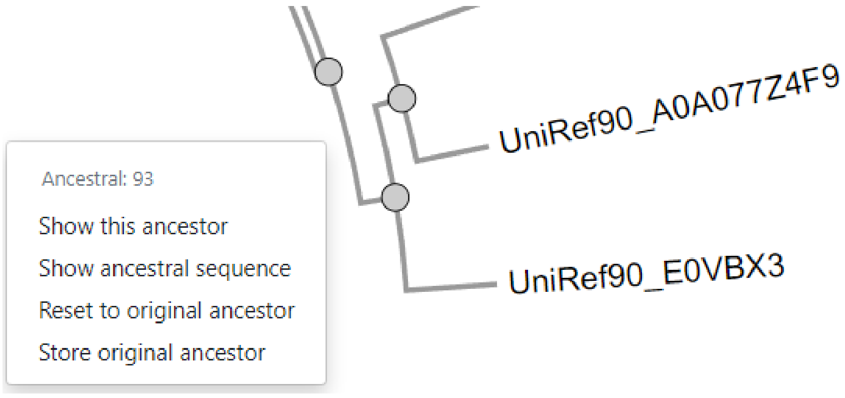


Figure 8.10: The options for a phylogenetic tree node on a result page.

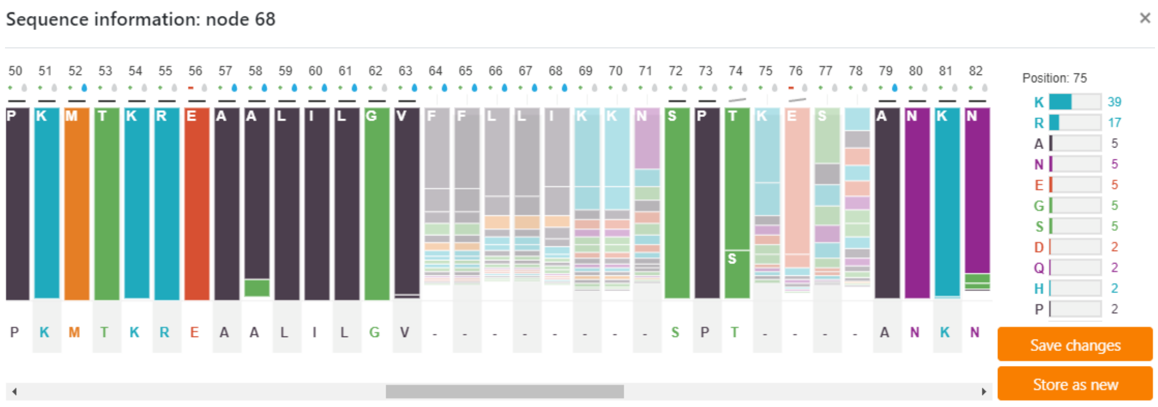
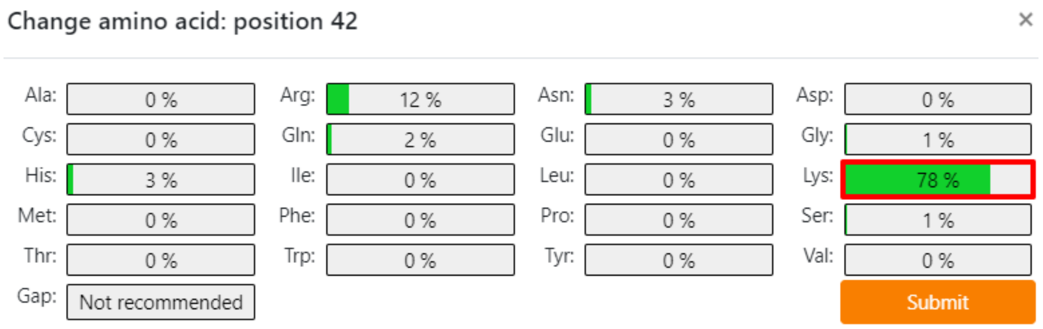


Figure 8.11: The sequence logo visualisation.



Change amino acid: position 42

Ala:	0 %	Arg:	12 %	Asn:	3 %	Asp:	0 %
Cys:	0 %	Gln:	2 %	Glu:	0 %	Gly:	1 %
His:	3 %	Ile:	0 %	Leu:	0 %	Lys:	78 %
Met:	0 %	Phe:	0 %	Pro:	0 %	Ser:	1 %
Thr:	0 %	Trp:	0 %	Tyr:	0 %	Val:	0 %
Gap:	Not recommended						Submit

Figure 8.12: A selection of an amino acid for a certain position.

Chapter 9

Conclusion

In the first theoretical chapter, proteins were described and classified. In the second chapter, the stability of proteins was discussed. The methods for the prediction of stable proteins were presented with an emphasis on a phylogenetic analysis. Ancestral sequence reconstruction (ASR) was mentioned as an example of the phylogenetic analysis. In the next chapters, the important bioinformatics algorithms, which are used in ASR, were described. It includes the algorithms and tools for searching for homologs in databases of sequences, described in the third theoretical chapter. In the next chapter, the methods for a sequence alignment were presented. The last theoretical chapter was dedicated to the algorithms for the construction of phylogenetic trees.

In the implementation chapter, an automated ASR tool called FireProt^{ASR} was presented. Firstly, the workflow of FireProt^{ASR} was described. The workflow consists of two parts and implements all the operations required for the reconstruction of ancestors from one query sequence, including homolog collection, construction of an alignment, construction of the phylogenetic tree and rooting of the tree. Then, FireProt^{ASR} was described from the implementation point of view. The architecture of FireProt^{ASR} consists of the computational core, backend and frontend. The strongest emphasis was on the implementation of the frontend, because it was implemented within this thesis together with the backend, while the core was only partly modified.

Afterwards, a successor prediction was described. It is a novel technique which extends the idea of ASR into the opposite evolutionary direction. Firstly, a parser of Newick files was mentioned. The successor analysis uses the results of the ASR as an input so it has to parse the phylogenetic tree in Newick format into a suitable internal representation. Then, the successor algorithm itself was presented. It computes an evolutionary trend for each position in the sequence, and extrapolates the value for the next evolutionary step.

In the following chapter, the results of a successor analysis were presented. The successors are consistent with the reconstructed ancestral sequences. The comparison of the successors with the ancestors suggests that the outputs are meaningful and valuable. The only issue could be in a gap prediction, which is a complex task for both the successor and ancestor reconstruction. Nevertheless, further laboratory experiments are necessary to thoroughly examine the beneficial effects of the successor analysis.

As a last part, the new user interface was presented. It provides a fast and intuitive way to control FireProt^{ASR} and display both the ancestor and successor sequences.

Bibliography

[1] *Britannica.* Encyclopaedia Britannica, 2019. [Online; accessed 11-04-2022]. Available at: https://www.britannica.com/science/transcription-genetics.

[2] Akanuma, S. Characterization of reconstructed ancestral proteins suggests a change in temperature of the ancient biosphere. *Life.* Multidisciplinary Digital Publishing Institute. 2017, vol. 7, no. 3, p. 33.

[3] Altschul, S. F. Generalized affine gap costs for protein sequence alignment. *Proteins: Structure, Function, and Bioinformatics.* Wiley Online Library. 1998, vol. 32, no. 1, p. 88–96.

[4] Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology.* Elsevier. 1990, vol. 215, no. 3, p. 403–410.

[5] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z. et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research.* Oxford University Press. 1997, vol. 25, no. 17, p. 3389–3402.

[6] Amdursky, N., Sepunaru, L., Raichlin, S., Pecht, I., Sheves, M. et al. Electron Transfer Proteins as Electronic Conductors: Significance of the Metal and Its Binding Site in the Blue Cu Protein, Azurin. *Advanced Science.* march 2015, vol. 2. DOI: 10.1002/advs.201400026.

[7] Anfinsen, C. B. Principles that govern the folding of protein chains. *Science.* JSTOR. 1973, vol. 181, no. 4096, p. 223–230.

[8] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N. et al. The protein data bank. *Nucleic acids research.* Oxford University Press. 2000, vol. 28, no. 1, p. 235–242. Available at: http://www.rcsb.org/.

[9] Biasini, M., Schmidt, T., Bienert, S., Mariani, V., Studer, G. et al. OpenStructure: an integrated software framework for computational structural biology. *Acta Crystallographica Section D: Biological Crystallography.* International Union of Crystallography. 2013, vol. 69, no. 5, p. 701–709.

[10] Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A. et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic acids research.* Oxford University Press. 2003, vol. 31, no. 1, p. 365–370.

[11] Branden, C. I. and Tooze, J. *Introduction to protein structure.* Garland Science, 2012.

[12] Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T. J. et al. Multiple sequence alignment with the Clustal series of programs. *Nucleic acids research.* Oxford University Press. 2003, vol. 31, no. 13, p. 3497–3500.

[13] Chinenov, Y. and Kerppola, T. Close encounters of many kinds: Fos-Jun interactions that mediate transcription regulatory specificity. *Oncogene.* may 2001, vol. 20, p. 2438–52. DOI: 10.1038/sj.onc.1204385.

[14] Coordinators, N. R. Database resources of the national center for biotechnology information. *Nucleic acids research.* Oxford University Press. 2016, vol. 44, D1, p. D7–D19.

[15] Darwin, C. *On the origin of species, 1859.* Routledge, 2004.

[16] Dayhoff, M., Schwartz, R. and Orcutt, B. 22 a model of evolutionary change in proteins. *Atlas of protein sequence and structure.* National biomedical research foundation Silver Spring, MD, USA. 1978, vol. 5, p. 345–352.

[17] Dehouck, Y., Grosfils, A., Folch, B., Gilis, D., Bogaerts, P. et al. Fast and accurate predictions of protein stability changes upon mutations using statistical potentials and neural networks: PoPMuSiC-2.0. *Bioinformatics.* Oxford University Press. 2009, vol. 25, no. 19, p. 2537–2543.

[18] Doig, A. J. and Sternberg, M. J. Side-chain conformational entropy in protein folding. *Protein Science.* Wiley Online Library. 1995, vol. 4, no. 11, p. 2247–2251.

[19] Edgar, R. C. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics.* BioMed Central. 2004, vol. 5, no. 1, p. 1–19.

[20] Edgar, R. C. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics.* Oxford University Press. 2010, vol. 26, no. 19, p. 2460–2461.

[21] Eftink, M. R. The use of fluorescence methods to monitor unfolding transitions in proteins. *Biophysical journal.* Elsevier. 1994, vol. 66, no. 2, p. 482–501.

[22] Eick, G. N., Bridgham, J. T., Anderson, D. P., Harms, M. J. and Thornton, J. W. Robustness of reconstructed ancestral protein functions to statistical uncertainty. *Molecular biology and evolution.* Oxford University Press. 2017, vol. 34, no. 2, p. 247–261.

[23] Ellis, R. J. and Vies, S. M. van der. MOLECULAR CHAPERONES. *Annual Review of Biochemistry.* 1991, vol. 60, no. 1, p. 321–347. DOI: 10.1146/annurev.bi.60.070191.001541. PMID: 1679318.

[24] Felsenstein, J. *The Newick tree format.* [Online; accessed 22-04-2022]. Available at: https://evolution.genetics.washington.edu/phylip/newicktree.html.

[25] Felsenstein, J. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of molecular evolution.* Springer. 1981, vol. 17, no. 6, p. 368–376.

[26] Ferdjani, S., Ionita, M., Roy, B., Dion, M., Djeghaba, Z. et al. Correlation between thermostability and stability of glycosidases in ionic liquid. *Biotechnology letters.* Springer. 2011, vol. 33, no. 6, p. 1215–1219.

[27] Fetrow, J. S. and Bryant, S. H. New programs for protein tertiary structure prediction. *Bio/technology.* Nature Publishing Group. 1993, vol. 11, no. 4, p. 479–484.

[28] Finn, R. D., Clements, J. and Eddy, S. R. HMMER web server: interactive sequence similarity searching. *Nucleic acids research.* Oxford University Press. 2011, vol. 39, suppl_2, p. W29–W37.

[29] Fitch, W. M. and Farris, J. S. Evolutionary trees with minimum nucleotide replacements from amino acid sequences. *Journal of Molecular Evolution.* Springer. 1974, vol. 3, no. 4, p. 263–278.

[30] Gibbs, A. J. and McIntyre, G. A. The diagram, a method for comparing sequences: Its use with amino acid and nucleotide sequences. *European journal of biochemistry.* Wiley Online Library. 1970, vol. 16, no. 1, p. 1–11.

[31] Göbel, U., Sander, C., Schneider, R. and Valencia, A. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics.* Wiley Online Library. 1994, vol. 18, no. 4, p. 309–317.

[32] Greenfield, N. J. Using circular dichroism spectra to estimate protein secondary structure. *Nature protocols.* Nature Publishing Group. 2006, vol. 1, no. 6, p. 2876–2890.

[33] Gromiha, M. M. *Protein bioinformatics: from sequence to function.* Academic press, 2010.

[34] Guharoy, M. and Chakrabarti, P. Conserved residue clusters at protein-protein interfaces and their use in binding site identification. *BMC bioinformatics.* Springer. 2010, vol. 11, no. 1, p. 1–17.

[35] Guindon, S. and Gascuel, O. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic biology.* Society of Systematic Zoology. 2003, vol. 52, no. 5, p. 696–704.

[36] Hanson Smith, V., Kolaczkowski, B. and Thornton, J. W. Robustness of ancestral sequence reconstruction to phylogenetic uncertainty. *Molecular biology and evolution.* Oxford University Press. 2010, vol. 27, no. 9, p. 1988–1999.

[37] Henikoff, S. and Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences.* National Acad Sciences. 1992, vol. 89, no. 22, p. 10915–10919.

[38] Henikoff, S. and Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences.* National Acad Sciences. 1992, vol. 89, no. 22, p. 10915–10919.

[39] Hillis, D. M., Moritz, C., Mable, B. K. and Olmstead, R. G. *Molecular systematics.* Sinauer Associates Sunderland, MA, 1996.

[40] Hon, J., Borko, S., Stourac, J., Prokop, Z., Zendulka, J. et al. EnzymeMiner: automated mining of soluble enzymes with diverse structures, catalytic properties and stabilities. *Nucleic acids research.* Oxford University Press. 2020, vol. 48, W1, p. W104–W109.

[41] Jäckel, C., Bloom, J. D., Kast, P., Arnold, F. H. and Hilvert, D. Consensus protein design without phylogenetic bias. *Journal of molecular biology.* Elsevier. 2010, vol. 399, no. 4, p. 541–546.

[42] Jin, G., Nakhleh, L., Snir, S. and Tuller, T. Maximum likelihood of phylogenetic networks. *Bioinformatics.* Oxford University Press. 2006, vol. 22, no. 21, p. 2604–2611.

[43] Jukes, T. H., Cantor, C. R. et al. Evolution of protein molecules. *Mammalian protein metabolism.* 1969, vol. 3, p. 21–132.

[44] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M. et al. Highly accurate protein structure prediction with AlphaFold. *Nature.* Nature Publishing Group. 2021, vol. 596, no. 7873, p. 583–589.

[45] Karlin, S. and Altschul, S. F. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences.* National Acad Sciences. 1990, vol. 87, no. 6, p. 2264–2268.

[46] Karsch Mizrachi, I., Takagi, T., Cochrane, G. and Collaboration, I. N. S. D. The international nucleotide sequence database collaboration. *Nucleic Acids Research.* Oxford University Press. 2018, vol. 46, D1, p. D48–D51.

[47] Katoh, K., Misawa, K., Kuma, K.-i. and Miyata, T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic acids research.* Oxford University Press. 2002, vol. 30, no. 14, p. 3059–3066.

[48] Kauzmann, W. *Sulfur in Proteins.* Elsevier, 1959.

[49] Kawashima, S., Ogata, H. and Kanehisa, M. AAindex: amino acid index database. *Nucleic acids research.* Oxford University Press. 1999, vol. 27, no. 1, p. 368–369.

[50] Kent, W. J. BLAT—the BLAST-like alignment tool. *Genome research.* Cold Spring Harbor Lab. 2002, vol. 12, no. 4, p. 656–664.

[51] Kimura, M. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution.* Springer. 1980, vol. 16, no. 2, p. 111–120.

[52] Kinene, T., Wainaina, J., Maina, S. and Boykin, L. Rooting trees, methods for. *Encyclopedia of Evolutionary Biology.* Elsevier. 2016, p. 489.

[53] Koonin, E. V. Orthologs, paralogs, and evolutionary genomics. *Annu. Rev. Genet.* Annual Reviews. 2005, vol. 39, p. 309–338.

[54] Kumar, M. S., Bava, K. A., Gromiha, M. M., Prabakaran, P., Kitajima, K. et al. ProTherm and ProNIT: thermodynamic databases for proteins and protein–nucleic acid interactions. *Nucleic acids research.* Oxford University Press. 2006, vol. 34, suppl_1, p. D204–D206.

[55] Lehmann, M., Kostrewa, D., Wyss, M., Brugger, R., D'Arcy, A. et al. From DNA sequence to improved functionality: using protein sequence comparisons to rapidly design a thermostable consensus phytase. *Protein engineering.* Oxford University Press. 2000, vol. 13, no. 1, p. 49–57.

[56] Lehninger, A. L., Nelson, D. L., Cox, M. M., Cox, M. M. et al. *Lehninger principles of biochemistry.* Macmillan, 2005.

[57] Leszczynski, J. F. and Rose, G. D. Loops in globular proteins: a novel category of secondary structure. *Science.* American Association for the Advancement of Science. 1986, vol. 234, no. 4778, p. 849–855.

[58] Levinthal, C. How to fold graciously. *Mossbauer spectroscopy in biological systems.* University of Illinois Press Urbana, IL. 1969, vol. 67, p. 22–24.

[59] Levitt, M. and Chothia, C. Structural patterns in globular proteins. *Nature.* Nature Publishing Group. 1976, vol. 261, no. 5561, p. 552–558.

[60] Lipman, D. J. and Pearson, W. R. Rapid and sensitive protein similarity searches. *Science.* American Association for the Advancement of Science. 1985, vol. 227, no. 4693, p. 1435–1441.

[61] Maddison, W. P., Donoghue, M. J. and Maddison, D. R. Outgroup analysis and parsimony. *Systematic biology.* Society of Systematic Zoology. 1984, vol. 33, no. 1, p. 83–103.

[62] Maguire, E., Rocca Serra, P., Sansone, S.-A. and Chen, M. Redesigning the Sequence Logo with Glyph-based Approaches to Aid Interpretation. In: *EuroVis (Short Papers).* 2014.

[63] Mahan, G. D. *Quantum mechanics in a nutshell.* Princeton University Press, 2008.

[64] Martínek, T. *Zarovnání sekvencí.* FIT VUT Brno. [pdf] Accessed: 11-04-2022.

[65] Mazurenko, S. Predicting protein stability and solubility changes upon mutations: data perspective. *Chem. Cat. Chem.* 2020, vol. 12.

[66] McNaught, A. D., Wilkinson, A. et al. *Compendium of chemical terminology.* Blackwell Science Oxford, 1997.

[67] Menardo, F., Loiseau, C., Brites, D., Coscolla, M., Gygli, S. M. et al. Treemmer: a tool to reduce large phylogenetic datasets with minimal loss of diversity. *BMC bioinformatics.* Springer. 2018, vol. 19, no. 1, p. 1–8.

[68] Mirny, L. and Shakhnovich, E. Evolutionary conservation of the folding nucleus. *Journal of molecular biology.* Elsevier. 2001, vol. 308, no. 2, p. 123–129.

[69] Modarres, H. P., Mofrad, M. and Sanati Nezhad, A. Protein thermostability engineering. *RSC advances*. Royal Society of Chemistry. 2016, vol. 6, no. 116, p. 115252–115270.

[70] Morrison, J. L., Breitling, R., Higham, D. J. and Gilbert, D. R. A lock-and-key model for protein–protein interactions. *Bioinformatics*. Oxford University Press. 2006, vol. 22, no. 16, p. 2012–2019.

[71] Musil, M., Khan, R. T., Beier, A., Stourac, J., Konegger, H. et al. FireProtASR: a web server for fully automated ancestral sequence reconstruction. *Briefings in bioinformatics*. Oxford University Press. 2021, vol. 22, no. 4, p. bbaa337.

[72] Musil, M., Konegger, H., Hon, J., Bednar, D. and Damborsky, J. Computational design of stable and soluble biocatalysts. *Acs Catalysis*. ACS Publications. 2018, vol. 9, no. 2, p. 1033–1054.

[73] Musil, M., Stourac, J., Bendl, J., Brezovsky, J., Prokop, Z. et al. FireProt: web server for automated design of thermostable proteins. *Nucleic acids research*. Oxford University Press. 2017, vol. 45, W1, p. W393–W399.

[74] Musil, M. *Computational Design of Stable Proteins*. Brno, CZ, 2021. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/phd-thesis/967/.

[75] Nagar, S. D. *Homology Terminology: Never Say the Wrong Word Again*. Accessed: 11-04-2022. Available at: https://bitesizebio.com/26762/homology-terminology-never-say-wrong-word/.

[76] Needleman, S. B. and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*. Elsevier. 1970, vol. 48, no. 3, p. 443–453.

[77] Nguyen, L.-T., Schmidt, H. A., Von Haeseler, A. and Minh, B. Q. IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular biology and evolution*. Oxford University Press. 2015, vol. 32, no. 1, p. 268–274.

[78] Notredame, C., Higgins, D. G. and Heringa, J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*. Elsevier. 2000, vol. 302, no. 1, p. 205–217.

[79] Overington, J. P., Al Lazikani, B. and Hopkins, A. L. How many drug targets are there? *Nature reviews Drug discovery*. Nature Publishing Group. 2006, vol. 5, no. 12, p. 993–996.

[80] Penel, S., Arigon, A.-M., Dufayard, J.-F., Sertier, A.-S., Daubin, V. et al. Databases of homologous gene families for comparative genomics. In: BioMed Central. *BMC bioinformatics*. 2009, vol. 10, no. 6, p. 1–13.

[81] Polizzi, K. M., Bommarius, A. S., Broering, J. M. and Chaparro Riggers, J. F. Stability of biocatalysts. *Current opinion in chemical biology*. Elsevier. 2007, vol. 11, no. 2, p. 220–225.

[82] Porter, L. L. and Looger, L. L. Extant fold-switching proteins are widespread. *Proceedings of the National Academy of Sciences.* National Acad Sciences. 2018, vol. 115, no. 23, p. 5968–5973.

[83] Rao, R. B., Fung, G. and Rosales, R. On the dangers of cross-validation. An experimental evaluation. In: SIAM. *Proceedings of the 2008 SIAM international conference on data mining.* 2008, p. 588–596.

[84] Rees, D. C. and Robertson, A. D. Some thermodynamic implications for the thermostability of proteins. *Protein Science.* Wiley Online Library. 2001, vol. 10, no. 6, p. 1187–1194.

[85] Reinhold Hurek, B. and Shub, D. A. Self-splicing introns in tRNA genes of widely divergent bacteria. *Nature.* Nature Publishing Group. 1992, vol. 357, no. 6374, p. 173–176.

[86] Ribeiro, A. J. M., Holliday, G. L., Furnham, N., Tyzack, J. D., Ferris, K. et al. Mechanism and Catalytic Site Atlas (M-CSA): a database of enzyme reaction mechanisms and active sites. *Nucleic acids research.* Oxford University Press. 2018, vol. 46, D1, p. D618–D623.

[87] Rost, B. Twilight zone of protein sequence alignments. *Protein engineering.* Oxford University Press. 1999, vol. 12, no. 2, p. 85–94.

[88] Russell, P. J. *IGenetics: A Molecular Approach.* Blackwell Science Oxford, 2009.

[89] Saitou, N. and Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution.* 1987, vol. 4, no. 4, p. 406–425.

[90] Sanderson, M. J. A nonparametric approach to estimating divergence times in the absence of rate constancy. 1997.

[91] Sankoff, D. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics.* SIAM. 1975, vol. 28, no. 1, p. 35–42.

[92] Schmucker, D., Clemens, J. C., Shu, H., Worby, C. A., Xiao, J. et al. Drosophila Dscam is an axon guidance receptor exhibiting extraordinary molecular diversity. *Cell.* Elsevier. 2000, vol. 101, no. 6, p. 671–684.

[93] Schultz, T. R. and Churchill, G. A. The role of subjectivity in reconstructing ancestral character states: a Bayesian approach to unknown rates, states, and transformation asymmetries. *Systematic Biology.* JSTOR. 1999, vol. 48, no. 3, p. 651–664.

[94] Sehnal, D., Bittrich, S., Deshpande, M., Svobodová, R., Berka, K. et al. Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research.* Oxford University Press. 2021, vol. 49, W1, p. W431–W437.

[95] Semple, C., Steel, M. et al. *Phylogenetics.* Oxford University Press on Demand, 2003.

[96] Service, R. F. The game has changed. AI triumphs at protein folding. *Science.* 2020, vol. 370, no. 6521, p. 1144–1145. DOI: 10.1126/science.370.6521.1144.

[97] Sevier, C. S. and Kaiser, C. A. Formation and transfer of disulphide bonds in living cells. *Nature reviews Molecular cell biology.* Nature Publishing Group. 2002, vol. 3, no. 11, p. 836–847.

[98] Shank, S. D., Weaver, S. and Kosakovsky Pond, S. L. Phylotree. js-a JavaScript library for application development and interactive data visualization in phylogenetics. *BMC bioinformatics.* Springer. 2018, vol. 19, no. 1, p. 1–5.

[99] Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular systems biology.* John Wiley & Sons, Ltd Chichester, UK. 2011, vol. 7, no. 1, p. 539.

[100] Smith, L. J., Fiebig, K. M., Schwalbe, H. and Dobson, C. M. The concept of a random coil: Residual structure in peptides and denatured proteins. *Folding and Design.* Elsevier. 1996, vol. 1, no. 5, p. R95–R106.

[101] Smith, T. F., Waterman, M. S. et al. Identification of common molecular subsequences. *Journal of molecular biology.* Elsevier Science. 1981, vol. 147, no. 1, p. 195–197.

[102] Smyth, M. and Martin, J. X Ray crystallography. *Molecular Pathology.* BMJ Publishing Group. 2000, vol. 53, no. 1, p. 8.

[103] Sokal, R. R. A statistical method for evaluating systematic relationships. *Univ. Kansas, Sci. Bull.* 1958, vol. 38, p. 1409–1438.

[104] Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics.* Oxford University Press. 2014, vol. 30, no. 9, p. 1312–1313.

[105] Tanford, C. Protein denaturation: Part C. Theoretical models for the mechanism of denaturation. *Advances in protein chemistry.* Elsevier. 1970, vol. 24, p. 1–95.

[106] Thompson, J. D., Higgins, D. G. and Gibson, T. J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research.* Oxford university press. 1994, vol. 22, no. 22, p. 4673–4680.

[107] Tria, F. D. K., Landan, G. and Dagan, T. Phylogenetic rooting using minimal ancestor deviation. *Nature Ecology & Evolution.* Nature Publishing Group. 2017, vol. 1, no. 1, p. 1–7.

[108] UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research.* Oxford University Press. 2021, vol. 49, D1, p. D480–D489.

[109] Valdar, W. S. Scoring residue conservation. *Proteins: structure, function, and bioinformatics.* Wiley Online Library. 2002, vol. 48, no. 2, p. 227–241.

[110] Whitford, D. *Proteins: Structure and Function.* Wiley, 2013. ISBN 9781118685723.

[111] Wijma, H. J., Floor, R. J. and Janssen, D. B. Structure-and sequence-analysis inspired engineering of proteins for enhanced thermostability. *Current opinion in structural biology.* Elsevier. 2013, vol. 23, no. 4, p. 588–594.

[112] Yang, B., Wang, Y. and Qian, P.-Y. Sensitivity and correlation of hypervariable regions in 16S rRNA genes in phylogenetic analysis. *BMC bioinformatics.* BioMed Central. 2016, vol. 17, no. 1, p. 1–8.

[113] Yang, Z. PAML 4: phylogenetic analysis by maximum likelihood. *Molecular biology and evolution.* Oxford University Press. 2007, vol. 24, no. 8, p. 1586–1591.

[114] Zhou, H.-X. and Pang, X. Electrostatic interactions in protein structure, folding, binding, and condensation. *Chemical reviews.* ACS Publications. 2018, vol. 118, no. 4, p. 1691–1741.

Appendix A

Content of DVD

The attached DVD contains the following files and directories:

- **fireprot-asr-backend**: Directory with source code of the backend.
- **fireprot-asr-core**: Directory with source code of the core.
- **fireprot-asr-frontend**: Directory with source code of the frontend.
- **latex**: Latex source code for the text of this thesis.
- **thesis.pdf**: The text of this thesis.