VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Magistro baigiamasis darbas

Baltymų struktūrų modeliavimas trimatėje erdvėje

(Modelling of proteins structures in three-dimensional space)

Atliko: 3 kurso studentas
Paulius Milmantas (parašas)
Darbo vadovas:
doc. dr. Linas Petkevičius (parašas)
Recenzentas:
vyres.m.d. dr. Kliment Olechnovič (parašas)

Vilnius
2024

Santrauka

Darbe yra tiriami galimi ColabFold programos, kuri praplečia AlphaFold2, rezultatų kokybės pagerinimo būdai. Vienas iš jų yra naujų sekų paieškos algoritmų praplėtimas. Dabartinis ColabFold sprendimas leidžia surasti sekas taikant tik MMSeq2 algoritmą. Sukurtas sprendimas leidžiantis lengvai pridėti savo nurodytus algoritmus ir taip praplėsti ColabFold programą. ColabFold darbe buvo praplėstas DIAMOND ir SWIPE algoritmais. Kitas pasiūlymas yra sujungti MMSeq2 rezultatus su vienu iš kitų baltymų paieškos algoritmų. Pridėtinius rezultatus galima filtruoti, taikant sekų suderinamumo indeksą, kuris yra apskaičiuojamas su mašininio mokymosi modelių ansamblių struktūra.

Raktiniai žodžiai: ColabFold, AlphaFold2, baltymų lankstymas, DIAMOND, SWIPE

Summary

This work explores possible ways to improve the quality of the ColabFold program results. One of them is the extension of new protein sequence search algorithms. Current ColabFold solution only allows sequence search using MMSeq2 algorithm. A solution is presented that allows to easily add specified algorithms and thus extend the ColabFold application. In this work ColabFold has been extended with the DIAMOND and SWIPE algorithms. Another proposal is to combine the results of MMSeq2 with one of the other protein search algorithms. The added results can be filtered using a sequence compatibility index, which is computed with an ensemble structure of machine learning models.

Keywords: ColabFold, AlphaFold2, protein folding, DIAMOND, SWIPE

Turinys

Įvadas

Baltymų modeliavimas trimatėje erdvėje yra reikalingas, norint sužinoti baltymo struktūrą ir jo funkcines savybes. Nuo to, kaip susilankstys baltymas 3D erdvėje, priklauso jo funkcionalumas [IG07]. Turint 3D baltymo struktūrą galima nuspėti, kaip baltymas reaguoja su kitais baltymais, kokios yra šio baltymo funkcinės savybės ir kiti požymiai. Šios žinios leidžia baltymą kontroliuoti ar netgi modifikuoti.

Baltymų susilankstymas yra sudėtingas procesas. Vienas iš populiariausių laboratorinių būdų rekonstruoti baltymą yra naudoti rentgeno spindulių kristalografiją [Ryu17]. Šis metodas leidžia eksperimentiškai aptikti apytikslę baltymo struktūrą. Naudojant rentgeno spindulius, yra apšvitinimas baltymas. Spinduliai išsisklaido į visas puses ir yra matuojama, kur išsisklaidė, kokiu intensyvumu. Pasitelkiant šį metodą, gaunami apytiksliai rezultatai, tačiau šis metodas yra labai brangus ir reikia gerai apmokytų žmonių [Dar21]. Šis metodas nėra vienintelis laboratorinis metodas, egzistuoja ir daugiau laboratorinių modeliavimo metodų, kaip „NMR“ ar „Cryo-EM“, kurie yra svarbūs, tačiau jie turi savo teigiamas ir neigiamas savybes, todėl negalima apibrėžti vieno modeliavimo metodo panaudojimo visoms situacijoms. Dėl brangumo, ilgo proceso ir kitų metodų apribojimų yra sparčiai vystoma baltymų kompiuterinio modeliavimo sritis. Kompiuteriniai algoritmai gali sumažinti baltymų modeliavimo kaštus, reikia mažiau įrangos ir procesas, palyginus su minėtais laboratoriniais metodais, yra trumpas.

Šiame darbo buvo bandoma patvirtinti pagrindinę darbo hipotezę - galima pagerinti esamus baltymų modeliavimo algoritmus, įskaitant ColabFold, kuris praplečia AlphaFold2 ir paverčia jį labiau prieinamą žmonėms, patobulinus daugybinio sekų palyginio (MSA) sudarymą. Šiame darbe yra pateikti du pasiūlymai, kurie modifikuoja pradinių duomenų surinkimo procesą. Skyriuje 3.1 pateiktas pasiūlymas praplėsti esamą ColabFold programą, leidžiant pridėti daugiau sekų paieškos algoritmų. Darbe pateikti rezultatai, rodantys, kad su tam tikromis sekomis algoritmai tokie kaip SWIPE ar DIAMOND su Uniref90 duomenų baze veikia geriau už MMSeq2. Kitas darbe pateiktas pasiūlymas yra vykdyti duomenų paiešką su MMSeq2 ir kitu pasirinktu algoritmu. Papildomo algoritmo duomenys gali būti filtruojami skaičiuojant suderinamumo indekso reikšmes ir paliekant tik labiausiai suderinamas sekas. Šios metrikos skaičiavimas yra pateiktas skyriuje 3.2.1. Šis pasiūlymas buvo išbandytas - sukurtas mašininio mokymosi modelių ansamblis, tačiau ColabFold rezultatų nepavyko pagerinti.

Šiame darbe yra aptariama ColabFold rezultatų kokybės gerinimo tematika. Remiantis bandymais, siekiama pagerinti pLDDT metrikas, kurios yra esminės ColabFold modelio įvertinimo priemonės. Šios metrikos tiesiogiai nenusako baltymo susilankstymo tikslumo, tačiau nurodo, kaip gerai AlphaFold modelis spėja baltymo atomų atstumus, atliekant prognozuojamą vietinį atstumo skirtumo testą („predicted Local Distance Difference Test“). Taip yra išvedama reikšmė nuo 0 iki 100, kuri nurodo baltymo susilankstymo klaidų tikimybę tam tikrame segmente. Kuo reikšmė yra didesnė, tuo baltymas yra labiau tikėtina gerai, be klaidų susilankstęs ir, kad modelio spėjimas dėl baltymo atomų atstumų yra tikėtinai teisingas. [Car23].

Iškeltas pagrindinis tikslas - pasiūlyti ir validuoti matematinius ir infrastruktūrinius sprendimus, siekiant pagerinti baltymų modeliavimo algoritmus, patobulinant daugybinio sekų palyginio (MSA) sudarymą. Darbe iškelti uždaviniai:

1. Išanalizuoti ColabFold veikimą.
2. Pateikti pasiūlymus, kaip galima patobulinti pradinių duomenų rinkinio sudarymo ColabFold algoritmą.
3. Įgyvendinti pateiktus pasiūlymus.
4. Palyginti įgyvendintų pasiūlymų grąžinamų rezultatų pLDDT metrikas su dabartiniu ColabFold sprendimu.

1 Literatūros apžvalga

1.1 Bendriniai baltymų modeliavimo principai

1.1.1 Baltymo sudėtis

Baltymus sudaro aminorūgščių sekos. Gamtoje yra aptiktos apie 500 skirtingos aminorūgštys, tačiau tik 20 iš jų sudaro baltymus, esančius žmogaus organizme. Šios aminorūgštys yra L-izomerai ir alfa-aminorūgštys [LM21].

Aminorūgštys tarpusavyje yra sujungtos peptidiniais ryšiais, ko pasekoje atsiranda kelios atskiros peptidų grandinės. Baltymus dažniausiai sudaro kelios peptidų grandinės, kurios susijungia į polipeptidus. Peptidai gali atlikti tam tikras biologines funkcijas [M10].

Visos aminorūgštys turi vienodas pagrindines struktūrines dalis [Red08]. Struktūros centre visada yra anglies atomas (C). Su centru jungiasi keturios struktūrinės dalys: aminogrupė (NH_2), vandenilio (H) atomas, rūgštinė karboksilo grupė ($COOH$), organinė R grupė, literatūroje dažnai minima, kaip šoninė grandis, kuri yra unikali kiekvienai aminorūgščiai. Bendroji aminorūgščių cheminė struktūrinė formulė yra pateikta pav. 1. Aminorūgščių struktūra atomų lygyje yra pateikta pav. 2.

Baltymų sekų lyginimui tarpusavyje, galima analizuoti ir kategorizuoti aminorūgštis pagal jų savybes. Viena iš jų yra poliškumas [Red08]. Poliškumas nurodo elektros krūvio pasiskirstymą. Poliškumo savybė yra klasifikuojama pagal R grupę. Aminorūgštis galima išskaidyti į tokias kategorijas:

1. Nepolinės aminorūgštys [Red08].

   Šiai grupei priklauso šios rūgštys - glicinas, alaninas, valinas, leucinas, izoleucinas, prolinas, fenilalaninas, metioninas, triptofanas. Šios aminorūgštys turi hidrofobiškumo savybę - reaguoja su vandeniu.

2. Polinės aminorūgštys [Red08].

   Šiai grupei priklauso šios rūgštys - serinas, cisteinas, treoninas, asparaginas, glutaminas.

3. Rūgštinės aminorūgštys

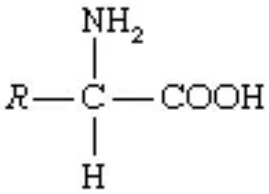
   Šiai grupei priklauso tik dvi aminorūgštys - asparto rūgštis ir glutamo rūgštis.

4. Bazinės aminorūgštys

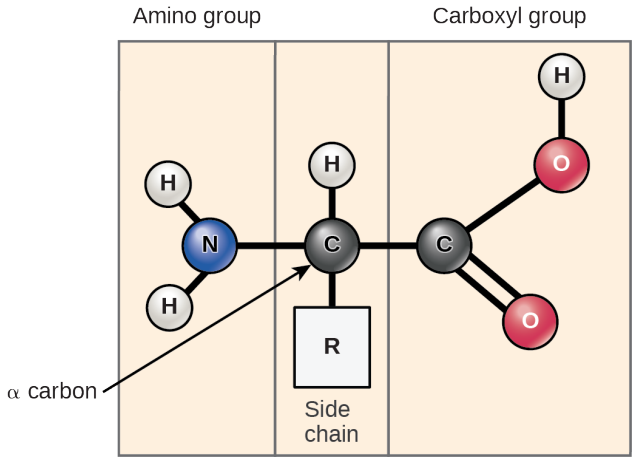
   Šiai grupei priklauso trys rūgštys - argininas, histidinas, lizinas.

1.1.2 Baltymų struktūros

Baltymai turi keturis struktūrinius lygmenis [SFB04]. Jie yra:



1 pav. Aminorūgšties atvaizdavimo pavyzdys [Red08]



2 pav. Aminorūgščių struktūros atvaizdavimas atomų lygio struktūroje [CCD18]

- Pirminis struktūrinis lygmuo - aminorūgščių seka, kuri sudaro baltymo polipeptidinę grandinę. Nuo šio struktūrinio lygmens reikšmės priklauso visi kiti struktūriniai lygmenys. Algoritmai, tokie kaip AlphaFold2, priima pirminę struktūrą tolimesnėms analizės žingsniams.
- Antrinis struktūrinis lygmuo - susilankstęs baltymas pagal egzistuojančius vandenilinius ryšius ir pagrindinę baltymo seką. Šį lygmenį sudaro α-spiralės, β-lakštai ir posūkiai.
- Tretinis struktūrinis lygmuo - susilanksčiusios atskiros polipeptidų grandinės. Visiems elementams yra priskiriamos koordinatės trimatėje erdvėje.
- Ketvirtinis struktūrinis lygmuo - pilnai susilankstęs baltymas - visos polipeptidų grandinės yra sujungiamos tarpusavyje ir gaunama pilna baltymo reprezentacija trimatėje erdvėje [SFB04].

AlphaFold 2 ir ColabFold algoritmai modeliuoja ketvirtinį struktūrinį lygmenį iš pirminio struktūrinio lygmens - aminorūgščių sekos. Tai yra atliekama praleidžiant antrinio ir tretinio lygmenų radinius.

1.2 AlphaFold2

1.2.1 Duomenų apdorojimas ir papildymas

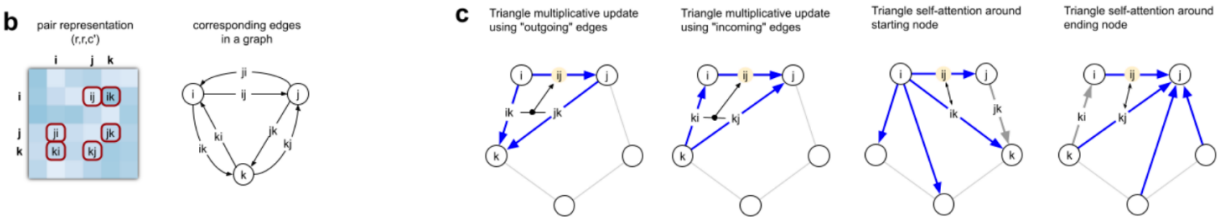
Pirmas algoritmo žingsnis - gauti daugiau informacijos apie duotą baltymo aminorūgščių seką [JEP+21]. Algoritmas paleidžia daug skirtingų procesų, kad išgautų panašias baltymų sekas įvairiose duomenų bazėse. Taip yra sugeneruojami daugybiniai sekų palyginiai (MSA - „multiple sequence alignment“) ir sekų šablonai. Turint panašių, evoliuciškai artimų baltymų sekų palyginį, galima prognozuoti, kurios baltymo sekos dalys turi didesnę mutavimo tikimybę. Baltymų struktūrų šablonai yra naudojami, nes nėra svarbu ar baltymas yra mutavęs ar nėra, tačiau struktūros šablonas išliks panašus.

1.2.2 Evoliucinis modelis

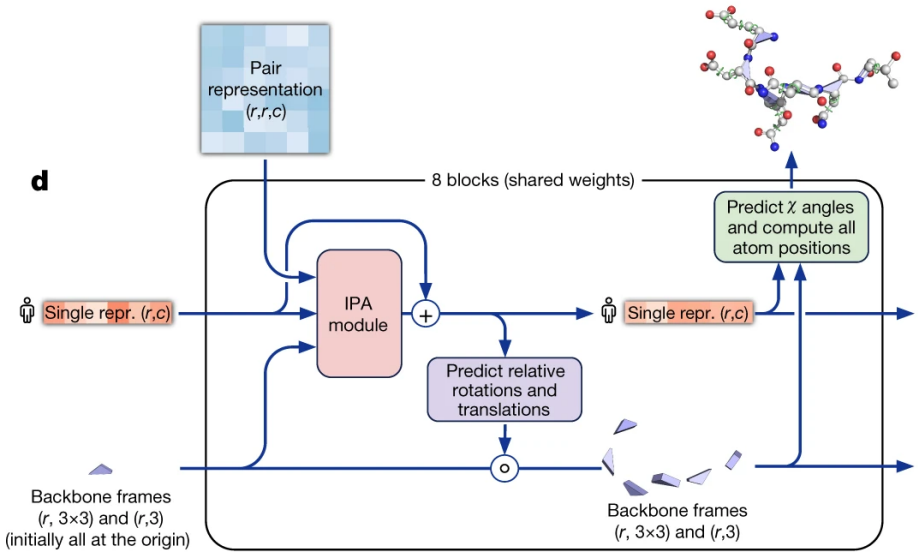
Evoliucinis modeliavimas AlphaFold2 algoritme išgauna informaciją apie turimus duomenis ir juos papildo [JEP+21]. AlphaFold2 algoritme ši dalis yra vadinama „Evoformer“. Išgavus papildomos informacijos, duomenys yra vėl siunčiami per evoliucinį modelį, kol yra išgaunama pakankamai papildomos informacijos apie baltymų palyginį. Ši informacija gali būti ir aptiktos koreliacijos tarp atitinkamų baltymų sekų. Aptikus tam tikras koreliacijas tarp sekų, kitos iteracijos metu galima aptikti dar daugiau koreliacijų, žinant apie egzistuojančias koreliacijas. Taip kelis kartus išanalizavus MSA ir baltymo šablonus, galima surinkti daug informacijos apie giminingas baltymų sekas. Šis modelis išsiskiria iš kitų neuroninių tinklų, skirtų baltymų analizei, iteratyviu duomenų tikslinimu. Kituose algoritmuose naudojami neuroniniai tinklai dažniausiai iš karto išveda tam tikrus rezultatus, kaip aproksimuotus atstumus, tačiau šis modelis yra skirtas duomenų tikslinimui ir naujų faktų gavimui.

Informacijos gavimas iš MSA ir šablonų yra vykdomas su transformerių tinklu [JEP+21]. „Evoformer“ dalyje yra realizuoti du transformerių modeliai: vienas yra skirtas MSA analizei, kitas šablonų analizei. Pirmas transformerių modelis, skirtas analizuoti MSA, apskaičiuoja dėmesį kiekvienai eilutei ir kiekvienam stulpeliui. Tai yra atliekama, nes analizuoti visą baltymų sekų matricą reikalautų per daug resursų. Dėmesio sutelkimui išgauti, naudojamas trikampių formos aminorūgščių išsidėstymas MSA struktūroje. Tokios trikampio struktūros formavimas yra pavaizduotas pav. 3. Šis analizės metodas skaičiuoja grafus tarp aminorūgščių ir efektyviai sukeičia atitinkamas MSA eilutes ar stulpelius. Galutinis dėmesio sutelkimo rezultatas yra modifikuotas MSA, kuris pavaizduoja, kurios aminorūgštys tarpusavyje yra labiausiai linkusios sąveikauti tarpusavyje.

Kitas transformerių tinklas yra skirtas šablonų analizei [JEP+21]. Šablonas - duomenys gauti iš įvairių baltymų duomenų bazių, pateikti PDB formatu. Šiame faile yra apibrėžtos kiekvieno baltymo atomo koordinatės trimatėje erdvėje. PDB faile taip pat pateikta bendrinė informacija apie analizuojamą baltymą - apibendrinimas, citavimas ir struktūrinės detalės [BWF+00]. Pirmasis transformeris analizuoja MSA duomenis - apskaičiuoja dėmesį kiekvienai eilutei ir kiekvienam stulpeliui. Rezultatai yra naudojami antro transformerio, analizuojant šablonus. Abiejų transformerių rezultatas - patobulinti pradiniai parametrai ir naujų faktų gavimas. AlphaFold2 algoritmo



3 pav. Aminorūgščių sąveikos paieška [JEP+21]



4 pav. Struktūrinis modulis [JEP+21]

apraše yra apibrėžta, kad „Evoformer“ atlieka 48 iteracijas tobulinant duomenis. Atlikus šias operacijas gaunamas rezultatas - baltymo atomų sąveikos modelis.

1.2.3 Struktūrinis modulis

Turint MSA matricą ir šablono struktūrą, galima pradėti generuoti baltymo struktūrą trimatėje erdvėje. Šį struktūros generavimą atlieka struktūrinis modulis [JEP+21]. Algoritmas kiekvieną baltymo atomą modeliuoja trikampio struktūroje - kiekviena aminorūgštis yra atvaizduojama trikampyje, išlaikant atstumus ir kampus tarp atomų, išlaikant sekos susilankstymą. Struktūrinis modulis kiekvienos iteracijos metu išveda transformacijų matricą, kuri apibrėžia, kaip turi būti transformuotos esamų baltymo atomų koordinatės trimatėje erdvėje. Transformacijoms generuoti yra naudojamas „nekintamo taško dėmesio“ modulis („Invariant Point Attention“ (IPA)). Struktūrinis modulis veiktų ir be IPA. Šis modulis yra skirtas pagerinti transformacijų matricų tikslumą, atsižvelgiant į turimus duomenis: baltymų sekų porų reprezentacijas, vieno baltymo sekos reprezentaciją ir sugeneruotą geometrinę baltymo sekos reprezentaciją trimatėje erdvėje. Modulis prognozuoja atomų koordinačių pokyčius nuo praeitos iteracijos ir grąžina modifikuotas sekų reikšmes. IPA išvestis ir parametrai pavaizduoti bendrame struktūrinio modulio algoritmo atvaizdavime, pateiktame pav. 4.

1.3 ColabFold

ColabFold yra naujas sprendimas pateiktas „Notebook“ formatu. Jis sujungia greitą sekų paieškos algoritmą MMseq2 su AlphaFold2. Šis sekų paieškos algoritmas MMseq2 yra 40-60 kartų greitesnis už AlphaFold2 naudojamus HMMer ir HHblits paieškos metodus. Tai yra pasiekta atlikus keletą modifikacijų. Viena iš jų yra ankstyvus prognozavimo stabdymo kriterijus [MSM$^+$22].

MMseq2 algoritmas, aprašytas [MSM$^+$22], yra sudarytas iš šių pagrindinių žingsnių:

1. „k-mer“ paieška. Turime posekį, kurio ilgis yra lygus k. Duomenų bazėse yra ieškomos tos sekos, kurios turi dvi panašius į duotą posekį sekas. Šis žingsnis svarbus, nes norima paimti tik tas sekas, kurios turi vertės galutiniam sekų rinkiniui. Turint tik daugiausiai potencialo turinčias sekas, yra sumažinamas reikalingų skaičiavimų skaičius kituose žingsniuose.

2. Vektorizuotas nesuderintas sulyginimas. Algoritmas sulygina sekas neįterpdamas tarpų. Tam yra atliekamos vektorinės operacijos, kad būtų galima lygiagrečiai vykdyti skaičiavimus ir juos pagreitinti. Skaičiavimai yra padalijami ir jiems priskiriami vektoriai, siekiant skaičiavimus paralelizuoti. Su kiekviena lyginama seka skaičiavimai yra išskirstomi į skirtingus procesoriaus branduolius.

3. Atliekamas išsamus sulyginimas tarp atrinktų potencialiai panašių sekų. „Smith-Waterman“ algoritmas yra dinaminio programavimo metodas, kuris leidžia įterpti tarpus į sekas taip, kad jos taptų vienodo ilgio. Tai leidžia gauti tikslesnį sulyginimą, nes sulyginamos sekos yra tiesiogiai palyginamos, nevertinant jų ilgio skirtumų.

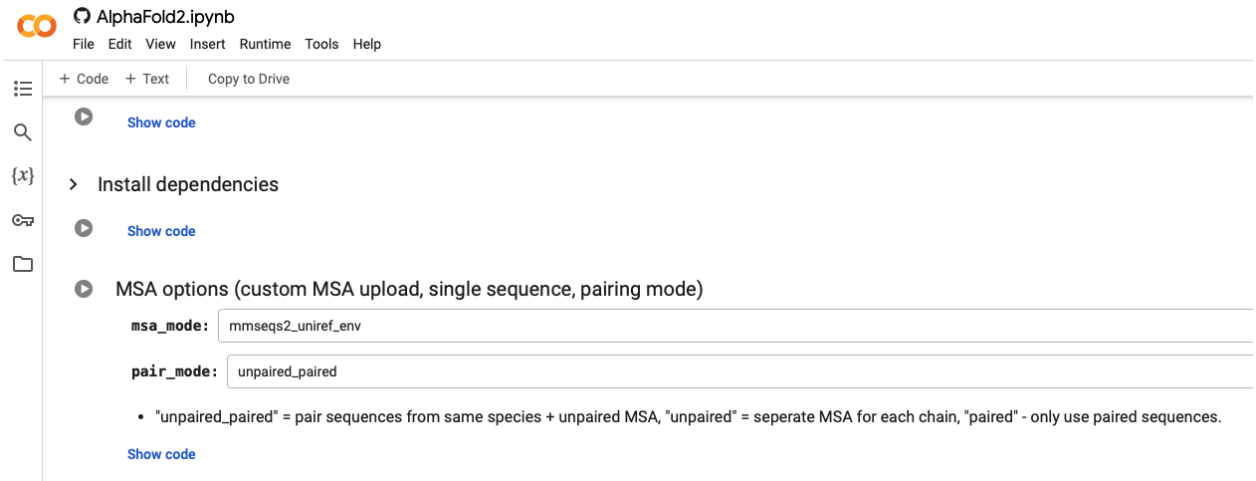
4. Filtravimas ir surūšiavimas. Po „k-merų“ paieškos ir vektorizuoto nesuderinto sulyginimo atliekamas rezultatų filtravimas ir surūšiavimas. Tai apima pašalinimą tokių potencialiai panašių sekų, kurios neatitinka tam tikrų kriterijų, pvz., panašumo lygio arba statistinių reikalavimų. Surūšiavimas leidžia atrinkti geriausius kandidatus tolimesnei analizei.

5. Geriausių atitikimų radimas. Šiame žingsnyje yra identifikuojami geriausi atitikimai tarp duotųjų ir potencialiai panašių sekų. Tai gali apimti rezultatų rūšiavimą pagal sulyginimo rezultatus, taip nustatant geriausius atitikimus pagal panašumo laipsnį arba kitus kriterijus.

2 Sistemos praplėtimo poreikis ir reikalavimai

ColabFold yra programa pateikta „Notebook“ formatu, kurią galima paleisti lokaliai naršyklėje arba pasinaudojant „Google Colab“ platforma ir jos teikiamais resursais. Ši programa pavertė baltymų lankstymą prieinamą visiems vartotojams – nereikia siųstis jokios programos, jokių baltymų duomenų bazių [SS17]. ColabFold, pradinių duomenų rinkinio MSA generavimui, pasiūlė taikyti MMSeq2 algoritmą, kuris geba lyginti pateiktą baltymo seką su keliomis duomenų bazėmis iš karto [SS17]. Pasiūlytas algoritmas yra daug kartų greitesnis už AlphaFold2 naudojamą sekų paieškos algoritmą, o tikslumas yra panašus. Atsisiųsti visas reikalingas baltymų duomenų bazes, reikalingas sekų lyginimui ir pradinių duomenų aibės sudarymui, naudojant ColabFold nėra būtina, nes pateiktas sprendimas susisiekia su egzistuojančiu galingu serveriu, kuris turi šias duomenų bazes ir atlieka visus reikalingus skaičiavimus, generuojant reikalingą duomenų aibę. Būtent šis serveris padaro baltymų analizę prieinamą visiems, nes nebėra reikalavimo turėti galingą įrangą ar siųstis didelius duomenų kiekius baltymų lyginimui.

ColabFold darbo pradžioje surenka pradinių duomenų rinkinį MSA, su kuriuo yra vykdomi sekantys žingsniai. Rinkinį surenka MMSeq2 algoritmas, patalpintas atskirame serveryje, todėl ColabFold pradinių duomenų rinkinio sudarymo algoritmo keitimas yra apsunkintas. Programa leidžia įkelti savo sudarytą MSA rinkinį, tačiau apsunkintas yra procesas, norint automatizuoti savo nurodytą duomenų surinkimo procesą ar dinamiškai pakeisti sekų atrankos algoritmą. ColabFold galimi pradinių duomenų surinkimo nustatymai yra pateikti pav. 5. Šiam sprendimui trūksta galimybės modifikuoti duomenų paieškos serverio nuorodą, algoritmo nustatymo, dinamiško duomenų bazės pridėjimo ir galimybės nurodyti integracijos autentifikavimo raktą. Ši galimybė, leisti ColabFold tiesiogiai komunikuoti su nurodytu serveriu, būtų ženklus vartotojo patirties pagerinimas, nes nereikėtų atlikti daug rankinių žingsnių, ypač, jei norima modeliuoti daug baltymų.



5 pav. Pradinių duomenų gavimo nustatymai

Šios problemos galėtų būti išspręstos sukuriant sprendimą, kuris būtų alternatyva dabartiniam MMSeq2 serveriui. Šis sprendimas turėtų veikti panašiai, kaip egzistuojantis serveris, tik priimtų

daugiau perduodamų parametrų, su kuriais būtų galima dinamiškai keisti pradinių duomenų sudarymo žingsnius. Surinkti pagrindiniai sprendimo funkciniai reikalavimai:

1. Sukurtas serveris turėtų būti patalpintas į konteinerį. Tai yra gera programavimo praktika, leidžianti programą paleisti „Kubernetes“ grupėse („klasteriuose“), bei lengvai plėsti programą horizontaliai ir vertikaliai.

2. Sprendimas turi būti pasiekiamas per RESTful programinės įrangos sąsają.

3. Užduočių eilių funkcionalumas. Sprendimas turi kiekvienai užduočiai priskirti eilės numerį ir turi būti galima patikrinti užduoties statusą, parsisiųsti rezultatus.

4. Užduotis turi galėti pateikti tik autorizuoti vartotojai, siekiant išvengti kenkėjiškos veiklos.

5. Užduotys turi turėti savo prioritetą. Duomenų surinkimo užduotys turi aukštą prioritetą, o užduotys, kaip duomenų bazių parsiuntimas, turi mažą prioritetą. Užduotis sustačius į eilę, pirma yra vykdomos aukšto prioriteto užduotys eilės tvarka, vėliau užduotys prioriteto mažėjimo tvarka.

6. Sukurtas sprendimas negali užsiblokuoti, vykdant ilgus skaičiavimus. Vartotojams turi būti ne ilgiau nei per sekundę, grąžinami sugeneruoti užduočių bilietai.

7. Sprendimas turi palaikyti paieškos algoritmus: „SWIPE“ ir „DIAMOND“.

8. Sprendimas turi leisti parsiųsti nurodytas duomenų bazes programos vykdymo metu ir baigus siuntimą, iš karto pradėti jas naudoti.

9. Sprendimas turi būti lengvai plečiamas naujais algoritmais, kurie yra patalpinti į atskirus konteinerius.

10. Pateikiant užduotis, sprendimas turi priimti papildomus parametrus, nusakančius pradinių duomenų sudarymo algoritmą ir naudojamą duomenų bazę.

Apibrėžus funkcinius reikalavimus, galima aiškiai apibrėžti kurių reikalavimų nėra MMSeq2 serveryje ir kuo siūlomas sprendimas yra pranašesnis:

1. 4-tas reikalavimas. MMSeq2 neturi vartotojų autorizacijos ir norint, galima pakenkti, siunčiant perteklinį nereikalingų užklausų srautą. Siūlomas sprendimas „docker-compose.yml“ faile turi nurodytus integracinius raktus ir yra aptarnaujamos tik tos užklausos, kurios turi minėtus raktus. Atsiradus šių raktų nutekėjimui, įgyvendintas papildomas apsaugos mechanizmas - ribojamas užklausų skaičius kiekvienam raktui. Su kiekvienu raktu galima įvykdyti tik 500 užklausų į serverį.

2. 5-tas reikalavimas. MMSeq2 užduotys yra vienodos - sudaryti sekų rinkinį. Dabartinis sprendimas naudoja papildomą užduočių tipą - duomenų bazių parsiuntimą. Šios užduotys yra itin ilgos, todėl tam yra reikalingas prioretizavimas.

3. 7-tas, 9-tas ir 10-tas reikalavimai. Dabartinis MMSeq2 serveris palaiko vienintelį sekų paieškos algoritmą. Siūlomas sprendimas yra geresnis, nes užduoties užklausoje galima nurodyti, kurį algoritmą reikia naudoti duomenų surinkimui.

4. 8-tas reikalavimas. Dabartinis sprendimas leidžia parsiųsti duomenų bazes tik prieš programos paleidimą, tačiau ne jos vykdymo metu.

Reikalinga taip pat praplėsti ColabFold programą, kad būtų geresnė vartotojo patirtis. Tam iškelti nauji reikalavimai:

1. „msa_mode“ parametras turi būti praplėstas naujais metodais. Šiuo pasirinkimu turi būti nurodomas tik algoritmas, neturi būti nurodoma duomenų bazė.

2. Pridėtas „server“ laukas, nurodantis duomenų paieškos serverio arba srauto paskirstytojo („load-balancer“) adresą.

3. Pridėtas „database“ laukas, kuris nurodytų naudojamą duomenų bazę.

4. Pridėtas „data_size“ laukas, nurodantis, koks maksimalus galimas MSA rinkinio dydis. Kai kurie algoritmai, tokie kaip SWIPE, priima šį parametrą ir tam tikrais atvejais jei jis yra per didelis, surandamos sekos su dideliais skirtumais, kurios nėra tinkamos lyginimui.

3 Sistemos praplėtimo sprendimai

Šiame skyriuje yra pateikiami sistemos praplėtimo sprendimai, kurių tikslas yra patvirtinti hipotezę, kuri teigia, kad galima pagerinti esamus baltymų modeliavimo algoritmus, tokius kaip ColabFold, patobulinus daugybinio sekų palyginio (MSA) sudarymą. Tikslą yra bandoma pasiekti netik sukūrus atskirą sprendimą ir gavus rezultatų palyginimą, tačiau ir kreipiant dėmesį į vartotojo patirtį ir naujo sprendimo naudojimo patogumą.

3.1 Sekų paieškos algoritmo keitimas

Analizuojant programą, buvo pastebėta, kad ji turi gerintinų vietų. Vienas šios programos taisytinas trūkumas - nėra leidžiama keisti sekų paieškos algoritmo ar lengvai pridėti savo duomenų bazės, kol programa yra jau paleista. ColabFold kūrėjai pateikė minėto serverio programinį kodą ir apibrėžė, kaip galima pridėti savo naują duomenų bazę, tačiau tai yra atliekama tik pasileidus savo nuosavą serverį ir tik prieš jo paleidimą [SS17].

Vienas iš galimų programos pagerinimų būdų yra praplėsti naudojamą MMSeq2 serverį, kuris yra atsakingas už pradinių duomenų generavimą. Šis praplėstas sprendimas galėtų priimti daugiau parametrų, pagal kuriuos būtų galima programai liepti keisti paieškos algoritmą į kitą. Pavyzdžiui, bandymo tikslams gali būti poreikis vykdyti pradinių duomenų generavimą, taikant „SWIPE“ ar „Diamond“ paieškos įrankius. Kitas galimas praplėtimas yra dinaminis duomenų bazės pridėjimas su nustatytais parametrais. Dabartinis sprendimas leidžia sukonfigūruoti serverį ir dirbti su nustatytomis duomenų bazėmis, tačiau tai nėra įmanoma programos vykdymo metu [SS17]. Tai galėtų būti atlikta pateikiant API adresą, į kurį būtų galima kreiptis, nurodant duomenų bazės adresą, jos dydį ir reikalingą duomenų bazės paruošimo metodą. Šis veiksmas sukurtų mažo prioriteto užduotį užduočių sekoje ir kai serveris nebus užimtas, duomenys galėtų būti parsiųsti ir paruošti tolimesniam darbui. Šis sprendimas taip pat turėtų būti sukurtas, taikant architektūrinius sprendimus, kurie leistų užduotis dėti į užduočių eilę, kai kurioms užduotims taikyti atitinkamą prioritetą, stebėti užduočių eilę ir prireikus lengvai praplėsti programinį kodą pageidaujamu algoritmu. Dabartinė pateikta serverio programa nėra šios architektūros ir jos negalima lengvai praplėsti.

3.1.1 Programų perkėlimas į konteinerius

Sprendimas yra sukurti apvalkalo architektūros programą, kuri naudotų pateiktus kitų programų konteinerius ir būtų lengvai plečiama. Pirmas šio sprendimo taikomas architektūrinis principas yra „Docker“ konteinerių naudojimas. Kiekviena nauja programa sekų lyginimui, būtų patalpinta atskiruose konteineriuose, siekiant, kad visą aplinką būtų galima lengvai sudiegti nepriklausomai nuo serverio operacinės sistemos ir būtų galima programą dinamiškai plėsti, padidinus resursų kiekį ir konteinerių kiekį. Tam kiekvienai naudojamai programai, kaip „SWIPE“ ar „Diamond“ buvo sukurti „Dockerfiles“ aprašai, apibrėžiantys kaip sukurti atitinkama konteinerį. Apraše 1 yra pateiktas „SWIPE“ programa, apraše 2 Diamond programa. Jame yra nurodyta reikalinga programos

paleidimo aplinka ir reikalingi žingsniai programos kompiliavimui. Taip kiekvienam konteineriui galime pateikti jam reikalingą aplinką atitinkamos programos kompiliavimui.

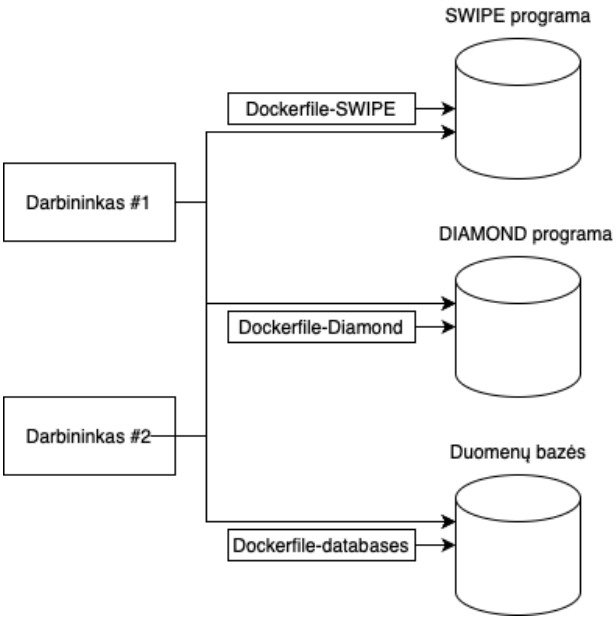
Aprašas 1: SWIPE Dockerfile

```
FROM --platform=linux/x86_64 gcc:13.2-bookworm
RUN apt-get update && apt-get install -y git
WORKDIR /app
RUN git clone https://github.com/PauliusMilmantas/swipe.git
WORKDIR /app/swipe
RUN make
```

Aprašas 2: DIAMOND Dockerfile

```
FROM --platform=linux/x86\_64 node:21-bookworm
WORKDIR /app
RUN wget
https://github.com/bbuchfink/diamond/releases/download/v2.1.9
/diamond-linux64.tar.gz
RUN tar -xzf diamond-linux64.tar.gz
```

Turint daug konteinerių, reikia juos sinchronizuoti – nurodyti paleidimo eigą ir nustatymus. Tam naudojamas „docker-compose“ aprašas. Jame yra aprašomi kiekvieno konteinerio tinklo prievadai, aplinkų kintamųjų reikšmės ir kietojo disko vietos - saugyklos, kurias konteineriai dalijasi. Bendra schema pateikta pav. 6. Darbininkai-servisai turi prieigą prie visų saugyklų. Kiekviena sekų lyginimo programa turi prieigą prie savo nuosavos saugyklos sukompiliuotiems failams saugoti ir prieigą prie bendros baltymų duomenų bazių saugyklos. Pirma integracijos aprašoma dalis yra pateikta apraše 3. Šioje dalyje yra aprašomas integracijos prievadas, su kuriuo komunikuoja ColabFold - teikiamos užduotys, kurioms gaunamas bilietas ir jį galima tikrinti. Ši integracijos dalis priklauso nuo žinučių perdavimo sistemos RabbitMQ, todėl pridėtas priklausomybės atributas, bei RabbitMQ prisijungimo duomenys. Apraše taip pat pridėti autorizacijos raktai, bei jų limitai. „dev“ raktas turi nurodytą 5000 kvietimų limitą, o „adminToken“ neturi limito. Limitas priskirtas aukštas, nes pateikus užduotį rezultatai yra tikrinami kas kelias sekundes. Kiekvienos užklausos metu sumažėja likusių kvietimų skaičius, todėl parinktas nemažas kvietimų skaičius, kad vartotojas galėtų sumodeliuoti 2-4 baltymus.



6 pav. Servisų struktūra

Aprašas 3: docker-compose Integracijos dalis

```
version: '3.8'
services:
  api:
    build:
      context: .
      dockerfile: Dockerfile-API
    ports:
      - "3000:3000"
    environment:
      - DEFAULT_TOKEN_LIMIT=5000
      - USER_TOKENS=["dev"]
      - ADMIN_TOKEN="adminToken"
      - RABBIT_USERNAME=admin
      - RABBIT_PASSWORD=admin
      - RABBIT_URL=rabbit
    depends_on:
      - rabbit
```

Apraše 4 yra pateikta darbininko sinchronizacijos dalis. Tai programos dalis, kuri vykdo visus skaičiavimus. Norint plėsti programą horizontaliai, reikėtų didinti šios programos dalies konteinerių skaičių. Ši dalis komunikuoja per žinučių perdavimo sistemą RabbitMQ su pagrindiniu programos prievadu „api“, todėl pateikti RabbitMQ prisijungimo duomenys ir nurodytas šios sistemos priklausomybės atributas. Ši programos dalis vykdo sekų paiešką, todėl reikalingos visos

turimos duomenų bazės, bei paieškos programų paleidimo failai, todėl yra užkraunamos duomenų saugyklos („volumes“), kuriose yra visos reikalingos duomenų bazės ir programos.

Aprašas 4: docker-compose Darbininko dalis

```
worker:
  build:
    context: .
    dockerfile: Dockerfile-Worker
  environment:
    - RABBIT_USERNAME=admin
    - RABBIT_PASSWORD=admin
    - RABBIT_URL=rabbit
  volumes:
    - swipe:/app/engines/swipe
    - diamond:/app/engines/diamond
    - databases:/app/databases
  depends_on:
    - rabbit
    - engine_swipe
    - engine_diamond
```

Apraše 5 yra pateikta žinučių perdavimo sistemos sinchronizacijos dalis. Ši sistema neturi priklausomybių nuo kitų sistemos dalių. Ji išsiskiria poreikiu saugoti duomenis diske. Tam reikia nurodyti vietas kietajame diske, kur turėtų būti rašomos užduotys, kurios dar nebuvo nuskaitytos, bei kur turi būti paliekami įrašai apie sistemos darbą.

Aprašas 5: docker-compose žinučių perdavimo dalis

```
rabbit:
  image: rabbitmq:3.12-management
  ports:
    - "5672:5672"
    - "15672:15672"
  volumes:
    - ~/.docker-conf/rabbitmq/data/:/var/lib/rabbitmq/
    - ~/.docker-conf/rabbitmq/log/:/var/log/rabbitmq
  environment:
    - RABBITMQ_DEFAULT_USER=admin
    - RABBITMQ_DEFAULT_PASS=admin
```

Apraše 6 yra pateiktos SWIPE ir Diamond programų sinchronizacijos dalys. Šios programos nuo kitų nepriklauso, joms reikia nurodyti tik vietas, kur yra saugomos duomenų bazės.

Aprašas 6: docker-compose paieškos programu dalis

```
engine_swipe:
  build:
    context: .
    dockerfile: Engines/SWIPE/Dockerfile
  volumes:
    - swipe:/app
engine_diamond:
  build:
    context: .
    dockerfile: Engines/DIAMOND/Dockerfile
  volumes:
    - diamond:/app
```

Apraše 7 yra apibrėžiamos naudojamos duomenų bazės ir programos - nurodomos jų vietos kietajame diske.

Aprašas 7: docker-compose duomenų bazės

```
volumes:
  engines:
  swipe:
  diamond:
  databases:
    driver: local
    driver_opts:
      o: bind
      type: none
      device: /Databases
```

3.1.2 Žinučių perdavimas

Sprendimas atliks daug skaičiavimų, kurie ilgai užtruks, todėl kol yra vykdomi vieni skaičiavimai, kito vartotojo užklausa negali būti apdorota. Šią problemą MMSeq2 serverio sprendimas išsprendžia, taikant bilietų sistemą. Vartotojas pateikia užklausą, gauna bilietą ir atitinkamu intervalu užklausia serverio ar atitinkama užduotis su nurodytu bilietu yra baigta [SS17]. Ši metodika pateiktame pasiūlyme yra taip pat naudojama. Tai yra įgyvendinta, taikant žinučių perdavimo sistemą „RabbitMQ“. Pagrindinis API servisas priima užduočių užklausas, priskiria joms po bilietą, išsaugo ir išsiunčia į užduočių eilę. Pagrindinis servisas ilgų užduočių neatlieka, todėl vartotojams

yra visada pasiekiamas ir atsako į HTTP užklausas. Iš užduočių eilės užduotis pasiima servisai-darbininkai. Jie analizuoja užklausas, atlieka ilgus skaičiavimus ir grąžina rezultatus į pagrindinį servisą per HTTP užklausą. Darbininkų skaičius gali būti lengvai didinimas, nes yra taikoma konteinerių ir žinučių perdavimo architektūra. Atitinkamos užklausos darbininkams, kaip duomenų bazės parsiuntimas, yra mažiau svarbios, todėl pateikiant kiekvieną užklausą į „RabbitMQ“ užduočių eilę, pridedamas užduoties atlikimo prioritetas. „RabbitMQ“ buvo pasirinkta būtent dėl šio reikalavimo, nes lyginant šį sprendimą su konkurentais, šis sprendimas turi įgyvendinęs prioritetų atributus žinutėse, o konkurentai, tokie kaip, „Apache Kafka“, neturi šio funkcionalumo [Rab23].

3.1.3 Naudojamos paieškų programos

Pasiūlyta architektūra yra lengvai plečiama ir galima nesunkiai pridėti įvairius sekų lyginimo programas. Kiekvienai programai tereikia aprašyti konteinerio sukūrimo žingsnius ir aprašyti metodą, kaip vykdyti naują programą, bei apdoroti jos rezultatus.

Viena iš kode jau aprašytų programų yra „SWIPE“. Tai yra įrankis skirtas sekų lyginimui ir jų lygiavimui. Įrankis tam naudoja „Smith-Waterman“ algoritmą [Rog11]. Pagal ColabFold autorius, šio metodo jautrumas baltymų sekoms yra labai panašus į MMseq2 [MSM+22]. Vėlesniuose tyrimuose buvo pastebėta, kad tam tikrais atvejais SWIPE įrankis yra jautresnis duomenims nei MMseq2 ar BLAST algoritmai [HSS16]. Iš literatūroje aprašyto bandymo rezultatų, galima padaryti išvadą, kad yra svarbu vartotojams leisti patiems pasirinkti naudojamą algoritmą, nes nėra vieno geriausio algoritmo, tinkančio visoms situacijoms. Šio algoritmo sudėtingumas yra $O(N^3)$ laiko atžvilgiu ir $O(N^2)$ atminties atžvilgiu. Įrankis naudoja algoritmo modifikacijas, leidžiančias pasiekti $O(N^2)$ laiko kompleksiškumą ir $O(N)$ atminties kompleksiškumą [Rog11].

3.1.4 Sekų aibės didinimas

Pradinis duomenų rinkinys ColabFold programoje yra sugeneruojamas, naudojant MMSeq2 algoritmą, kuris leidžia vienu metu lyginti pateiktą seką su keliomis duomenų bazėmis kartu [SS17]. Paprasčiausias būdas padidinti duomenų kiekį yra pridėti naują duomenų bazę. Pateiktas sprendimas leidžia pridėti duomenų bazes dinamiškai, nurodytam metodui, ir leidžia didinti pradinę duomenų aibę pasirinktomis duomenų bazėmis. Tokie įrankiai, kaip SWIPE, kurie nėra skirti darbui su keliomis duomenų bazėmis vienu metu, gali būti paleisti tiek kartų, kiek yra nurodyta patikrinti skirtingų duomenų bazių. Rezultatai gali būti agreguoti. Pateiktas sprendimas yra patalpintas į docker konteinerius ir naudoja architektūrą, kuri leidžia dinamiškai keisti darbininkų-servisų skaičių. Tai reiškia, kad padidinus darbininkų konteinerių skaičių iki naudojamų duomenų bazių skaičiaus, skaičiavimai gali būti paraleliškai vykdomi ir neužtrukti ilgiau nei skaičiavimų vykdymas su viena duomenų baze.

Sekų aibę galima didinti, sujungiant kelis naudojamus įrankius tarpusavyje. Jie gali analizuoti tas pačias duomenų bazes, tačiau jų išvestis gali būti skirtingi sekų poaibiai. Juos galima apjungti kartu, palikus tik unikalias sekas. Tai gali būti atliekama, kreipiantis į sprendimą kelis kartus, užda-

vus kelias skirtingas užduotis ir laukiant, kol jos bus išspręstos. Gavus rezultatus reikia prafiltruoti unikalias sekas. Taip gaunamas didesnis pradinių duomenų rinkinys.

3.1.5 Apibendrinimas

Pateikto sprendimo architektūrą sudaro kelios dalys. Apibendrinta architektūros schema yra pateikta pav. 7. Pirma yra pagrindinė API sąsaja. Jos servisas priima užklausas baltymų analizei, sukuria užduočiai bilietą ir išsiunčia užduotį į „RabbitMQ“ žinučių perdavimo sistemą. Vartotojai gali nustatytu intervalu kreiptis į šį servisą ir tikrinti užduoties statusą. Servisas yra patalpintas į Docker konteinerį.

Kitas komponentas yra „RabbitMQ“ žinučių perdavimo sistema, kuri leidžia pagrindinei API sąsajai, generuojančiai bilietus užduotims, perduoti darbą ir toliau aptarnauti vartotojų HTTP užklausas.

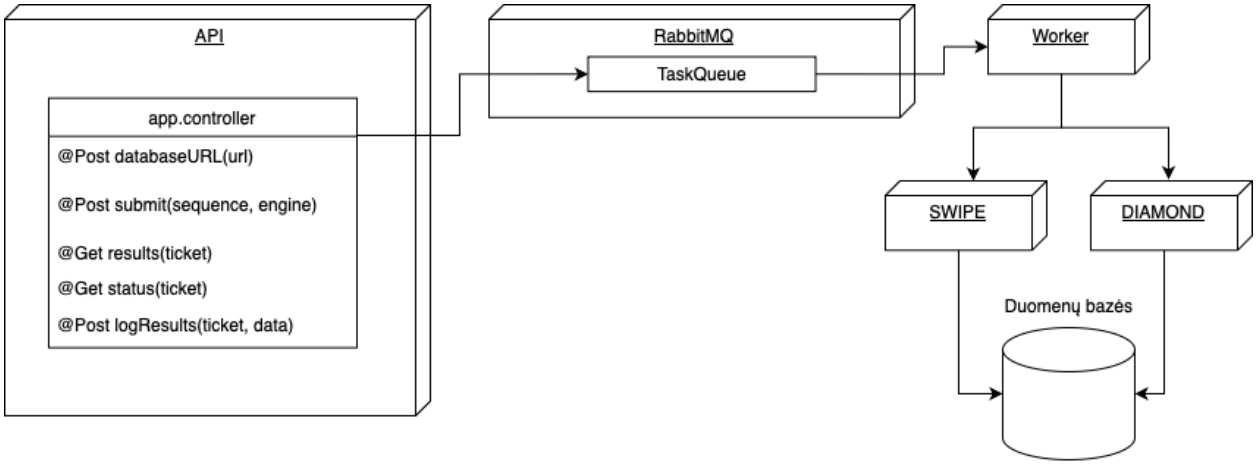
Paskutinis servisas-darbininkas skaito užduotis iš žinučių perdavimo sistemos ir vykdo užduotis. Šis servisas turi prieigą prie sukompiliuotų programų, kaip SWIPE ar Diamond, ir prireikus atitinkamą programą paleidžia su atitinkamais parametrais, priklausomai nuo užduoties. Servisui taip pat suteikta prieiga prie paruoštų duomenų bazių. Šis servisas yra patalpintas į Docker konteinerį ir tai leidžia lengvai plėsti šį sprendimą – didinti darbininkų-servisų skaičių. Jie gali dirbti nepriklausomai, dėl naudojamos žinučių perdavimo sistemos. Baigus darbą darbininkas-servisas pateikia HTTP užklausą pagrindiniam API servisui ir vartotojas tikrindamas bilietą gali pastebėti pasikeitusį užduoties statusą, parsisiųsti rezultatus.

Turint sukurtą sprendimą, reikia jį sujungti su ColabFold programa. Pirmas žingsnis yra paleisti pateiktą sprendimą lokaliai arba serveryje. Paleidus sprendimą lokaliai reikia parsiųsti duomenų bazes į nustatytą vietą ir jas sukonfigūruoti pagal naudojamų įrankių reikalavimus. Paleistas sprendimas gali nebūti pasiekiamas iš išorinio tinklo, todėl reikia atverti tinklo prievadus. Įprastai tai yra atliekama pakeitus maršrutizatoriaus nustatymus, pateikiant sprendimą paleidusio įrenginio vidinį IP adresą ir naudojamą prievadą. Taip sprendimas gali būti sukonfigūruotas išoriniam naudojimui ir į jį gali kreiptis ColabFold programa. Antras žingsnis – kreiptis į naują API sąsają iš ColabFold programos. Prieduose yra pateikti suprogramuoti sprendimai ir modifikuota ColabFold programa, kuri tai gali atlikti.

3.1.6 Rezultatai

Sukurtas sprendimas pateiktas prieduose. Sprendimas susideda iš dviejų dalių:

- Serverio programos, kuri yra patalpinta į konteinerius ir komunikuoja HTTP žinutėmis, pradinių duomenų generavimui.
- Modifikuotas ColabFold notebook failas, kuris, turi naujus pasirinkimus, leidžiančius pasirinkti pradinių duomenų generavimo metodą. Šis modifikuotas notebook failas kreipiasi į



7 pav. Apibendrinta struktūra

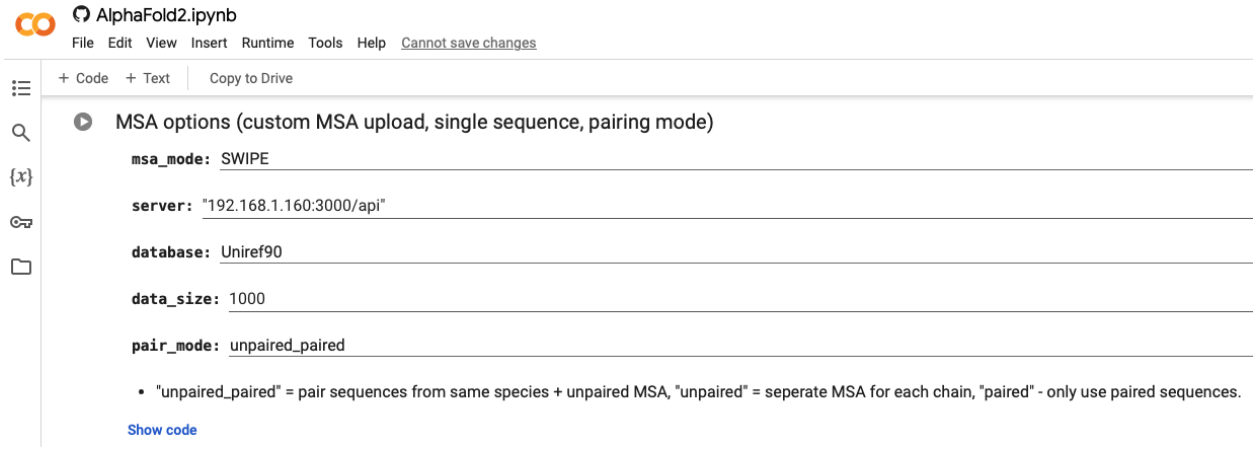
nurodytą serverį ir užduoda pradinių duomenų generavimo užduotis. Modifikacija pavaizduota pav. 8. Nauji laukai:

- „Database“ - naujas laukas, leidžiantis pasirinkti vieną iš kelių duomenų bazių, su kuria turėtų būti sukuriamas pradinių duomenų rinkinys. Šis laukas yra perduodamas HTTP užklausoje į naują nurodytą serverį.
- „Server“ - naujas laukas skirtas nurodyti serverio adresą, kuriame yra pasiekiama pradinio duomenų generavimo programa.
- „data_size“ - naujas laukas skirtas nurodyti naujai atrenkamų duomenų kiekį. Pav. 8 ši reikšmė yra nurodyta 1000. Tai reiškia, kad pasirinktas algoritmas atrinks labiausiai tinkamų 1000 sekų iš nurodytos duomenų bazės. Šios sekos yra papildomai agreguojamos prie MMSeq2 sugeneruotų duomenų. Duomenys yra agreguojami, nes nevykdant agregacijos, su tam tikrais parametrais, programos rezultatai yra ženkliai prastesni.
- „msa_mode“ - papildytas papildomais galimais algoritmais. T.y. SWIPE ir Diamond.

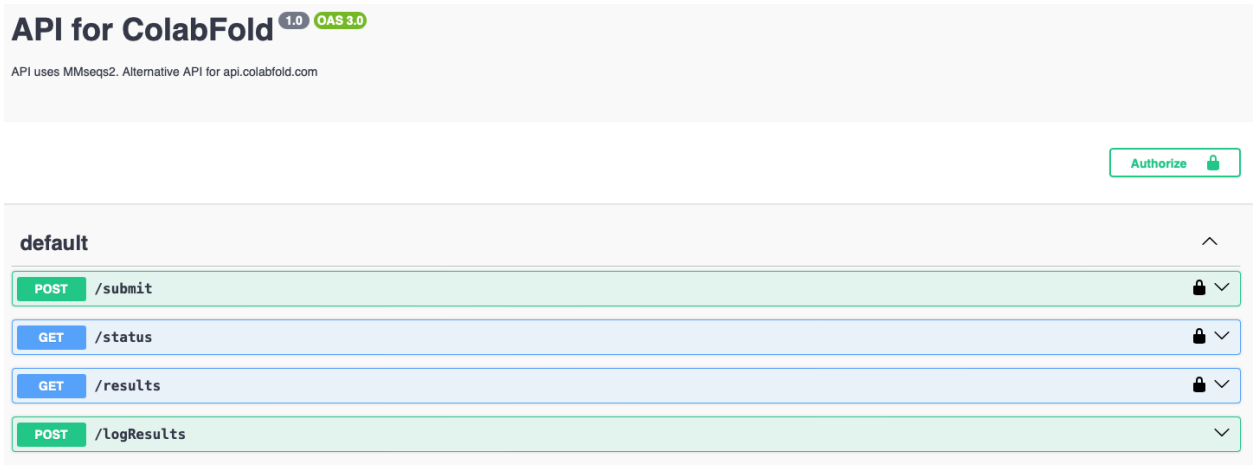
Sukurtas sprendimas yra prieinamas „OpenAPI“ standartu grįsta prieiga, kuri yra pateikta pav. 9.

Pavieniai algoritmai SWIPE ir DIAMOND, analizuojantys tik „Uniref90“ duomenų bazę, pateikė duomenų rinkinius, kurie nėra užtektini sumodeliuoti baltymus, tačiau pasinaudojus įgyvendintu nauju sprendimu ir sujungus kelių algoritmų duomenis, analizuojant „Uniref90“, buvo gauti rezultatai pateikti 1 lentelėje. Šiame bandyme buvo analizuota ColabFold faile jau pateikta seka „PIAQIHILEGRSDEQKETLIREVSEAISRSLDAPLTSVRVIITEMAKGHFGIGGELASK“, kuri turi aukštą pLDDT metriką. Rezultatuose matomas labai mažas 0,1% pLDDT kokybės metrikos padidėjimas, taikant MMSeq2 ir DIAMOND atrinktų duomenų agregavimą. Taikant MMSeq2 ir SWIPE duomenų sujungimą, rezultatų kokybės metrika yra lygiai tokia pati, kaip ir MMSeq2 algoritmo, nors SWIPE sugeneravo didelį duomenų rinkinį.

Sekantis bandymas buvo išanalizuoti seką „MKHGAALGAAK-



8 pav. ColabFold modifikacija



9 pav. „OpenAPI“ standartu grįsta prieiga

CHGKKHGLMGHHHGHLGMHHGHHMGKMACGAGMHLGAHKGGKC-LACLLLCCKGCGKLGCMLLHAGGHHHGLALGGAHMCMMKHAGKCCK“. Su šia seka MMSeq2 algoritmas grąžina prastesnius rezultatus. Rezultatai pateikti 2 lentelėje. MMSeq2 ir SWIPE agreguotų duomenų kokybės metrika pLDDT yra 51,1. Susilankstymo skirtumas pateiktas pav. 10. Taikant duomenų sujungimą su SWIPE, pLDDT kokybės metrika padidėjo 29,5%. DIAMOND algoritmas nesugebėjo rasti jokių sekų Uniref90 duomenų bazėje, todėl rezultatai išliko identiški, lyginant su MMSeq2.

3.2 Sekų aibės keitimas

3.2.1 Suderinamumo indeksas

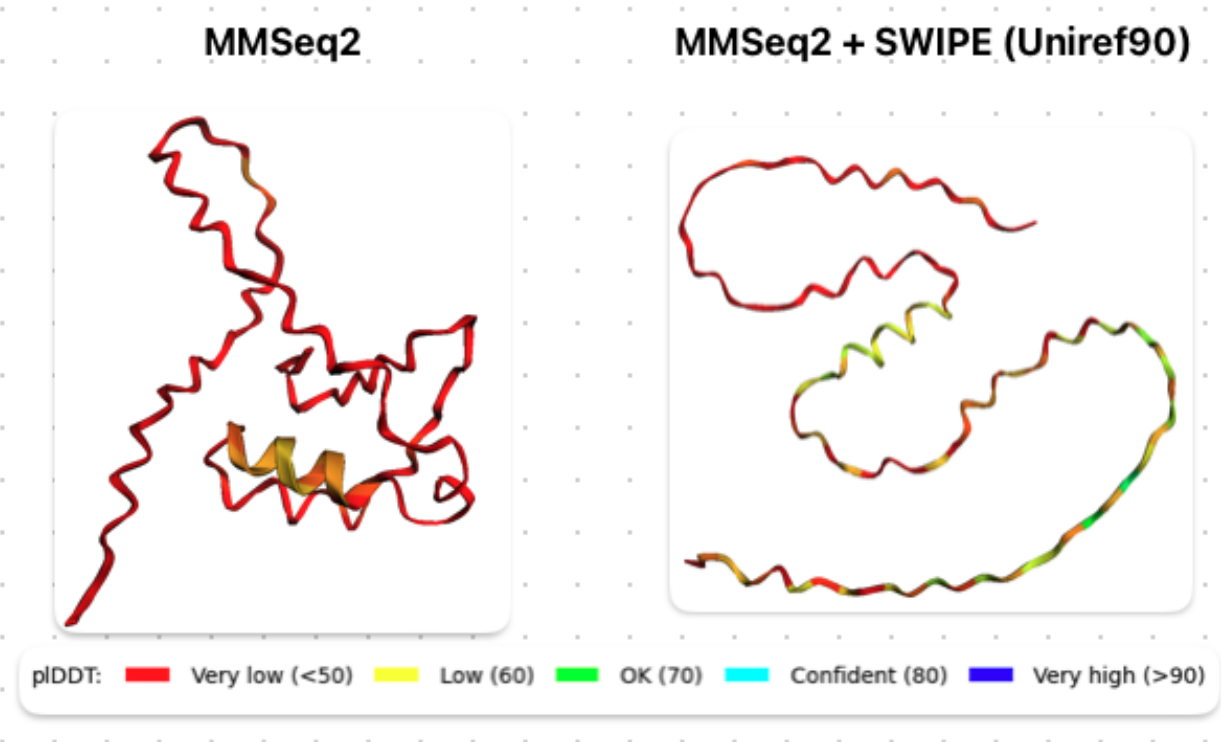
Pradinio sekų rinkinio - MSA sudarymą galima tobulinti, kad algoritmas efektyviau analizuotų baltymus. Šiame skyriuje yra pateiktas naujas eksperimentinio pobūdžio sprendimas, kurį galima lengvai toliau plėsti. Algoritmas yra grįstas mašininio mokymosi ansamblių architektūra. Kiekvienas mašininio mokymosi modelis ansamblių architektūroje naudoja skirtingai transformuotus

Algoritmas	Didžiausia pLDDT reikšmė
MMSeq2	97,4
MMSeq2	97,4
MMSeq2	97,4
MMSeq2 + DIAMOND	97,5
MMSeq2 + DIAMOND	97,5
MMSeq2 + DIAMOND	97,6
MMSeq2 + SWIPE	97,4
MMSeq2 + SWIPE	97,4
MMSeq2 + SWIPE	97,4

1 lentelė. ColabFold pirmas bandymas

Algoritmas	Didžiausia pLDDT reikšmė
MMSeq2	43,3
MMSeq2	43,3
MMSeq2	43,3
MMSeq2 + DIAMOND	43,3
MMSeq2 + DIAMOND	43,3
MMSeq2 + DIAMOND	43,3
MMSeq2 + SWIPE	56,1
MMSeq2 + SWIPE	56,1
MMSeq2 + SWIPE	56,1

2 lentelė. ColabFold antras bandymas



10 pav. Rezultatų palyginimas

duomenis ir prognozuoja koreliacijas tarp palyginio sekų rinkinio ir galutinio AlphaFold išvesties tikslumo, modeliuojant baltymus. Šis sprendimas kiekvienai sekai rinkinyje priskiria suderinamumo indekso (SI) reikšmę, o su šia reikšme, tolimesniam programos tobulinimui, pateikti tolimesni žingsniai skyriuje 3.2.2.

SI skaičiavimas yra pateiktas formulėje 1. Išvestis $SI \in [0; 100] \in \mathbb{R}$. Šioje formulėje yra sumuojama daugyba tarp svorių indeksų aibės $p \in [0; 100] \in \mathbb{R}$ ir ν - reikšmių aibės, nurodančios kiekvieno neuroninio tinklo išvestį. ν reikšmių aibė yra gaunama formulėje 2. Reikmės ν aibėje nurodo neuroninio tinklo prognozuojamą tikimybę, kad sekos tarpusavyje yra susijusios. Kuo ν aibėje esanti reikšmė yra didesnė, tuo didesnė tikimybė, kad sekos yra suderinamos tarpusavyje. Reikšmės p aibėje nurodo neuroninių tinklų svorius. Modelis, kuris grąžina tikslius rezultatus ir yra reikšmingas galutiniam rezultatui, turi aukštą svorį. Modelis, turintis mažą tikslumą arba mažą reikšmingumą galutiniam rezultatui - mažą svorį. Ši pasirinkta mašininio mokymosi tinklų architektūra, leidžianti priskirti svorius modeliams, leidžia įterpti nekokybiškus mašininio mokymosi modelius ir nekokybiškas transformacijų funkcijas su mažais svoriais ir neturėti didelės įtakos galutiniams rezultatui. Pasirinkta architektūra leidžia eksperimentuoti su įvairiomis transformacijų funkcijomis ir mašininio mokymosi modelių variacijomis, plečiant bendrą struktūrą ir nekenkiant galutiniam tikslumui.

$$SI = \sum_{n=1}^{|\nu|} \nu_n p_n \tag{1}$$

Formulėje 2 yra apskaičiuojamos reikšmės, siunčiant transformuotas baltymų sekas per sukurtus modelius. Kiekvienai egzistuojančiai transformacijos funkcijai turi egzistuoti atskiras mašininio mokymosi modelis arba turi būti nurodyta, kurį egzistuojantį modelį naudoti. Šioje funkcijoje NN - modelių aibė, δ - transformuotas baltymų sekų matricos.

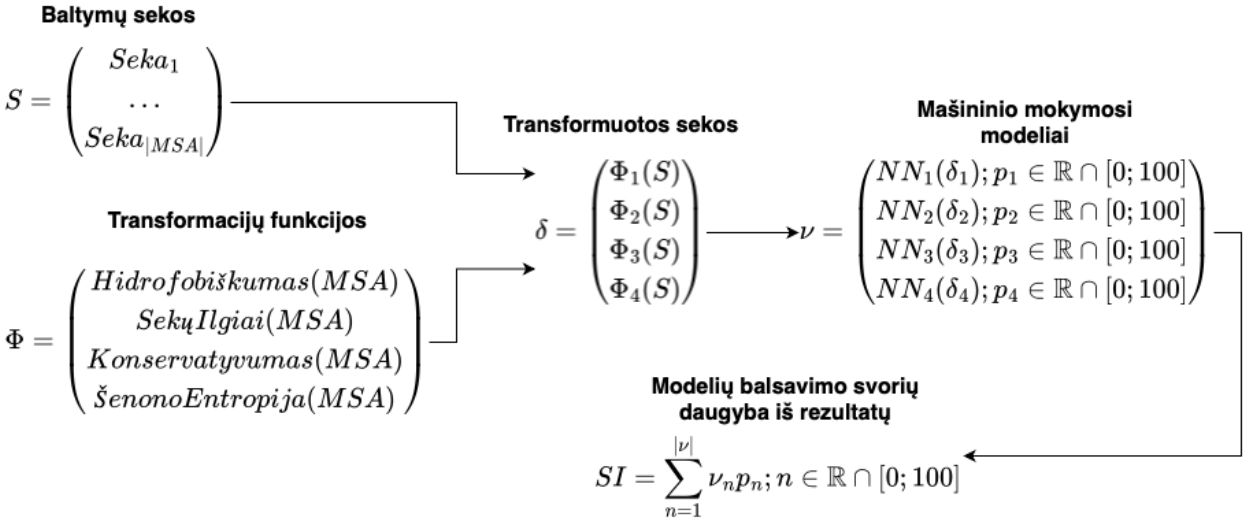
$$\nu = [NN_n(\delta_n) : n \in [1; |\delta|] \in \mathbb{N}] \tag{2}$$

Formulėje 3 yra apskaičiuojama aibė δ. Tai yra aibė transformuotų baltymų sekų. Kiekvienai egzistuojančiai transformacijos funkcijai, sukuriama nauja transformuota sekų matrica δ aibėje. Šioje formulėje Φ yra transformacijų funkcijų aibė.

Apskaičiuota δ reikšmė yra matrica, kurios kiekviena reikšmė yra atskira matrica, kurios reikšmė yra baltymų sekos, kurioms yra pritaikoma $n - oji$ transformacijos funkciją. Matricų kiekis δ matricoje priklauso nuo transformacijos funkcijų skaičiaus. Kiekvienos atskiros matricos dydis priklauso nuo baltymų sekų skaičiaus.

$$\delta = [\Phi_n(S) : n \in [1; |\Phi|] \in \mathbb{N}] \tag{3}$$

Skaičiavime lieka vienintelė nenustatyta mašininio mokymosi modelių svorių aibė p. Jos nustatymas vykdomas tyrimo pabaigoje. Pridėjus naują modelį modelių ansamblių architektūroje, reikia iš naujo perskaičiuoti modelių svorių aibę p. Prieš SI skaičiavimą, p aibė bus inicijuota iš



11 pav. SI skaičiavimo diagrama

$|N|$ reikšmių, kurios visos yra lygios $\frac{1}{|N|}$. Norint gauti tikslesnį SI, p reikšmės bus atitinkamai keičiamos tikrinant „AlphaFold“ rezultatus. p aibės keitimas, siekiant geriausių rezultatų, bus vykdomas, naudojant stochastinio gradientinio nuolydžio algoritmą, nes parametrų dimensija nedidelė, o stochastinio gradientinio nuolydžio užtenka išspręsti lokalių ekstremumų problemą.

Skaičiavimo SI eiga yra supaprastintai pavaizduota grafike 11, parodant duomenų kelią, funkcijų rezultatų išvestis.

Pridėjus naują transformacijos funkciją į Φ aibę ir naują modelį į NN aibę, kiekvieną kartą turės būti perskaičiuojamos reikšmės p aibėje. Tam naudojamas gradientiniu nuolydžiu grįstas neuroninis tinklas, kurio išvestis yra nauja p aibė.

Darbe pasiūlytas pagrindinis algoritmas yra pateiktas 1 algoritmo schemoje, taip pat pav. 11. Algoritmo pradiniai duomenys yra baltymų sekos ir norimo rinkinio dydis. Algoritmas naudoja 6 baltymų transformacijų funkcijas sekų meta-duomenų išgavimui. Kiekviena ši funkcija turi savo sukurtą mašininio mokymosi modelį, kuris kiekvienai sekai išveda suderinamumo indeksą SI. Kiekvienas modelis taip pat turi savo balsavimo svorį, kuris nulemia, kiek galutiniam rezultatui atitinkamas modelis turi įtakos. Apskaičiuojamos turimų modelių išvestys ir kiekvieno modelio išvestis atitinkamai padauginama iš modelio turimo apibrėžto svorio. Sekos yra išdėstomos suderinamumo indekso mažėjimo tvarka ir parenkamas tik nurodytas kiekis sekų su didžiausiomis reikšmėmis.

Darbe taip pat yra pasiūlytas alternatyvus algoritmas, pateiktas 2 algoritmo schemoje. Jis priima baltymų sekas, pageidaujamą sekų aibės dydį ir bendrą mašininio mokymosi modelį. Jis yra bendras visoms transformacijų funkcijoms ir todėl priima daugiau parametrų. Suderinamumo indekso skaičiavimas yra paprastesnis - jį kiekvienai sekai išveda modelis, nereikia jokių duomenų agreguoti, atsisakoma ansamblių architektūros.

Algoritmas 1 Siūlomas algoritmas

procedure MSAPARUOŠIMAS($sekos, aibesDydis$)
    $transformacijuFunkcijos \leftarrow$

        [$hidrofobiskumo()$, modelis, modelioSvoris],
        [$sekuAtstumu()$, modelis, modelioSvoris],
        [$funkcinioReiksmingumo()$, modelis, modelioSvoris],
        [$sekuIlgio()$, modelis, modelioSvoris],
        [$konservatyvumo()$, modelis, modelioSvoris],
        [$senono()$, modelis, modelioSvoris]; Funkcijos pateiktos 3.2.3 skyriuje

    $si \leftarrow$Masyvas inicijuotas nulinėmis reikšmėmis, skirtas saugoti kiekvienos sekos suderinamumą
    for $funkcija, i$ **in** $transformacijuFunkcijos$ **do**
        for $seka$ **in** $sekos$ **do**
            $si[i] = si[i] + funkcija[1](funkcija[0](s)) \cdot funkcija[2]$
        end for
    end for
    $siSekos \leftarrow$Tuščias masyvas saugoti sekas ir jų suderinamumą
    for $seka, i$ **in** $sekos$ **do**
        $siSekos[i] \leftarrow [seka, si[i]]$
    end for
    $siSekos \leftarrow siSekos.sort((a, b) => a[1] > b[1])$
    return $siSekos[: aibesDydis]$
end procedure

Algoritmas 2 Siūlomas alternatyvus algoritmas

procedure MSAPARUOŠIMAS($sekos, aibesDydis, bendrasModelis$)
    $transformacijuFunkcijos \leftarrow$

        $hidrofobiskumo()$,
        $sekuAtstumu()$,
        $funkcinioReiksmingumo()$,
        $sekuIlgio()$,
        $konservatyvumo()$,
        $senono()$

    $meta \leftarrow$Tuščias masyvas skirtas saugoti kiekvienos sekos transformacijų rezultatus
    for $funkcija, i$ **in** $transformacijuFunkcijos$ **do**
        for $seka$ **in** $sekos$ **do**
            $meta[i] = funkcija(seka)$
        end for
    end for
    return $bendrasModelis(sekos, meta)[: aibesDydis]$
end procedure

3.2.2 Sekų aibės mažinimas

Sekų aibės didinimas, pridedant papildomas duomenų bazes ar sujungiant kelių įrankių rezultatus, yra aprašytas skyriuje 3.1.4. Sekų aibę galima taip pat praplėsti, taikant apskaičiuotas suderinamumo indekso reikšmes. Turint apskaičiuotą pradinį duomenų rinkinį su MMSeq2 metodu ir pritaikius atitinkamą įrankį, tokį kaip SWIPE, papildžius duomenų rinkinį, gali gautis ir blogesni rezultatai. Vienas iš siūlymų, ColabFold programoje pridėti naują lauką - „Duomenų filtravimas pagal SI“. Šį lauką pasirinkus, gauti sugeneruoti papildomi duomenys iš pasiūlyto algoritmo būtų prafiltruojami - paliktos tik tos sekos, kurių SI vertė yra didesnė nei nurodyta riba.

Duomenų praplėtimas gali turėti neigiamos įtakos rezultatams, nes turint pradinių duomenų rinkinį, kuris yra gaunamas iš įvairių duomenų bazių, AlphaFold nusprendžia kiek sekų bus paimta iš šio MSA rinkinio prognozių kūrimui. T.y. pasirenkamas kiekis sekų, reikalingų prognozavimui ir jos yra atsitiktinai atrenkamos iš pradinio gauto duomenų rinkinio. Taip kiekvieną kartą, kai vykdome AlphaFold programą ir modeliuojame tą pačią seką, dažnai yra gaunami skirtingi rezultatai, nes atrinktų duomenų aibės dydis yra įprastai mažesnis už pradinių duomenų rinkinio dydį, todėl išvesties tikslumas yra nedeterministiškas.

3.2.3 Transformacijų funkcijos duomenų išgavimui

Turint aminorūgščių seką, neatlikus jokios analizės yra sunku pasakyti, kokias savybes baltymas turi [MSM+22]. Transformacijų funkcijos yra skirtos iš pradinio sekų rinkinio MSA išgauti meta-duomenis, pagal kuriuos būtų galima lyginti sekas. Iš šių meta-duomenų, mašininio mokymosi modeliai galėtų prognozuoti suderinamumo indeksą. Tolimesniuose skyreliuose yra aprašytos šios funkcijos.

3.2.3.1 Hidrofobiškumo transformacija

Meta-duomenų gavimui, galima pasitelkti Ponnuswamy hidrofobiškumo indeksavimą [YCZW10]. Ponnuswamy kiekvienai aminorūgščiai priskyrė hidrofobiškumo reikšmę [Pon91]. Turint aminorūgščių seką, iš apibrėžtos baltymo sekos galima kiekvienai aminorūgščiai surasti jos reikšmę, jas visas sudėti ir gautą sumą padalinti iš aminorūgščių skaičiaus sekoje [Pon91]. Aminorūgščių hidrofobiškumo reikšmės yra pateiktos lentelėje 3.

Turint būdą išgauti meta-duomenis, buvo bandyta patikrinti šio pavienio metodo efektyvumą. Buvo sugeneruota keletas pradinių duomenų rinkinių ir su kiekvienu paleista ColabFold programa (pasirinktas nustatymas įkelti MSA pačiam). Gauti rezultatai - susilankstymo kokybės metrika pLDDT su kiekvienu rinkiniu buvo fiksuojami excel faile. Jame buvo fiksuojama:

1. Eiliškumas sekų rinkinyje.
2. Sekos aprašas - klasifikuojamas kaip meta-duomenys modelyje.
3. Baltymo aminorūgščių seka.

Aminorūgštis	Hidrofobiškumo indeksas
K	5.72
N	6.17
D	6.18
E	6.38
P	6.64
Q	6.67
R	6.81
S	6.93
T	7.08
G	7.31
A	7.62
H	7.85
W	8.41
Y	8.53
F	8.99
L	9.37
M	9.83
I	9.99
V	10.38
C	10.93

3 lentelė. Hidrofobiškumo indeksai [YCZW10]

4. 20 stulpelių, apibrėžiančių, kiek ir kokių aminorūgščių yra pačioje sekoje.

5. Hidrofobiškumo reikšmė. Ji yra gaunama sudedant kiekvienos aminorūgšties reikšmę, pateiktą hidrofobiškumo konstantų lentelėje 3 ir padalijant iš aminorūgščių skaičiaus sekoje.

Turint šiuos duomenų stulpelius, sukurti modeliai, kurių tikslumas pateiktas lentelėje 4. Tiksliausiai prognozuoja jungtiniai modeliai. Tiksliausias modelis - jungtinis stochastinio gradiento nuolydžio modelis, sujungtas su tiesinės regresijos modeliu (taikant Ridge L2 duomenų normalizavimą). Modeliams sukurti buvo naudojamos 311 duomenų eilutės (70%), modelių testavimui 133 eilutės (30%).

Atlikus koreliacijų paiešką pagal Spirmano ir Pirsono metrikas, kurių skaičiavimas pateiktas formulėse 4 ir 5, nebuvo rasta jokių statistiškai reikšmingų koreliacijų.

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i}{n(n^2 - 1)} \tag{4}$$

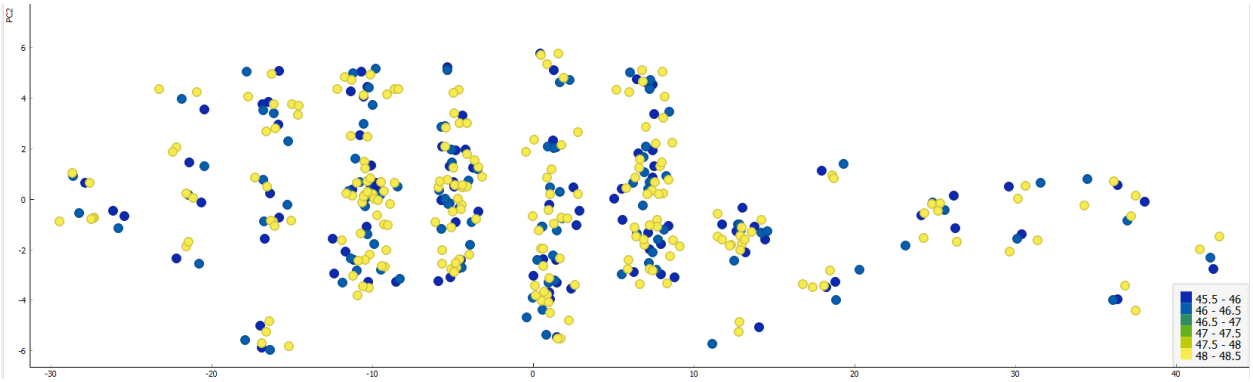
$$r = \frac{\sum_{i=1}^{n}(x_i - ex)(y_i - ey)}{\sqrt{\sum_{i=1}^{n}(x_i - ex)^2 \sum_{i=1}^{n}(y_i - ey)^2}} \tag{5}$$

Koreliacijas taip pat buvo bandoma atrasti, taikant pagrindinių komponentų analizę („Principal component analysis“ - PCA) Pav. 12 yra atvaizduojamas pirmų 2 komponentų grafikas, kurie turi 86% padengiamumą. Jis yra pateiktas lentelėje 5. PCA, turi aukštą dviejų komponentų padengia-

Modelis	MSE	RMSE	MAE	R2
Neuroninis tinklas	81,251	9,014	4,673	-38,879
Atsitiktinis miškas	3,340	1,828	1,747	-0,640
AdaBoost	5.596	2.365	2.090	-1.746
SGD	3626352448.36	60219.20	18033.18	-1779874863.34
Tiesinė regresija	4.889	2.211	1.9667	-1.399
Jungtinis (SGD, TR**)	1.735	1.317	1.246	0.148
Jungtinis (GB, AT**)	1.472	1.213	1.113	0.277

4 lentelė. Mašininio mokymosi modeliai tiriant hidrofobiškumą

** SGD - Stochastinis gradiento nuolydis, TR - tiesinė regresija taikant Ridge L2 duomenų normalizavimą, GB - gradientų sustiprinimas, AT - atsitiktinis miškas



12 pav. Pagrindinių komponentų analizė

mumą, tačiau nebuvo rasti požymiai, kurie parodytų koreliaciją tarp prognozių tikslumo ir PCA komponentų.

PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
0.82	0.04	0.02	0.02	0.01	0.01	0.01	0.009	0.008

5 lentelė. Pagrindinių komponentų padengiamumas

Taikant daugiamatį mastelio keitimą („Multidimensional scale modeling“ - MDS), apskaičiuotos reikšmės grafikas yra pateikiamas pav. 13. Skaičiavimui buvo naudojami Manhatano atstumai, kurių skaičiavimas yra pateiktas formulėje 6.

$$d = |x_2 - x_1| + |y_2 - y_1| \tag{6}$$

3.2.3.2 Sekų ilgio lyginimas

Nagrinėjant sekų ilgius, buvo pastebėta, kad jie gali nulemti, kiek seka yra informatyvi. Ilgesnės sekos turi daugiau potencialo susilankstyti - turint daugiau aminorūgščių sekoje, yra daugiau galimybių baltymui susilankstyti. Trumpesnės sekos turi mažiau galimybių susilankstyti. Sekos

13 pav. Daugiamatis mastelio keitimas, taikant Euklido atstumo skaičiavimą

ilgis ne visada lemia, ar ji daugiau susilankstys ar mažiau, tačiau bandymuose buvo pastebėta ši tendencija.

3.2.3.3 Konservatyvumas

Konservatyvumo skaičiavimo algoritmas MSA sekų rinkiniui leidžia nustatyti, kiek dažnai tam tikra aminorūgštis pasitaiko kiekvienoje pozicijoje tarp skirtingų sekų. Algoritmo schema yra pateikta diagramoje 3. Ši apskaičiuota reikšmė apibrėžia rinkinio konservatyvumą. Rezultatų pavyzdys yra pateiktas pav. 14. Pavaizduotame MSA rinkinyje sekos turi po 14 aminorūgščių. Šis algoritmas neatsižvelgia į sekų ilgio skirtumus, nes įprastai MSA būna sulygiuotas, todėl gali reikėti atlikti papildomą normalizaciją, jei sekų ilgiai skiriasi. Pagrindiniai algoritmo žingsniai:

1. Pradžioje yra nustatomas sekų skaičius (n) ir kiekvienos sekos ilgis (m) MSA struktūroje.
2. Tada inicijuojamas masyvas (c), kurio ilgis yra (m) nustatomas nulinėmis reikšmėmis.
3. Einama per kiekvieną poziciją (i) nuo 1 iki (m).
    - Kiekvienai pozicijai (i), sukuriama tuščia HashMap struktūrą ($counts$).

- Tada einama per kiekvieną seką (j) nuo 1 iki (n).
  - Nustatoma reikšmė (r), pozicijoje (i) sekoje (j).
  - Jei (r) nėra ($counts$) HashMap struktūroje, pridedama (r) prie ($counts$) su reikšme 1.
  - Jei (r) yra ($counts$), padidinama ($counts[r]$) reikšmę vienetu.
- Galiausiai priskiriama ($c[i]$) reikšmę lygią ($counts[i]$) padalijus iš (n).

4. Algoritmas grąžina masyvą (c), kuris dabar atspindi konservatyvumo reikšmes kiekvienai pozicijai MSA struktūroje.

Algoritmas 3 MSA konservatyvumo skaičiavimas

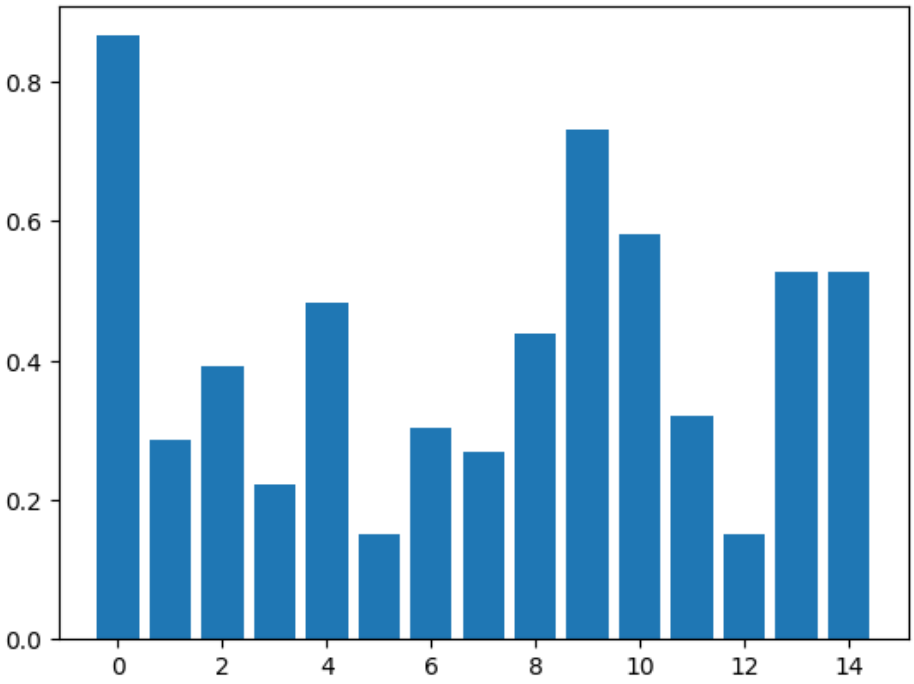
```
1:  procedure KONSERVATYVUMAS(MSA)
2:      n ← Sekų skaičius MSA struktūroje
3:      m ← Kiekvienos sekos ilgis MSA struktūroje
4:      c ← Masyvas, inicijuotas nulinėmis reikšmėmis, kurio ilgis lygis m
5:      for i ← 1 to m do
6:          counts ← Tuščia HashMap struktūra
7:          for j ← 1 to n do
8:              r ← Reikšmė i pozicijoje j sekoje
9:              if r not in counts then
10:                 counts[r] ← 1
11:             else
12:                 counts[r] ← counts[r] + 1
13:             end if
14:         end for
15:         c[i] ← counts[i] / n
16:     end for
17:     return c
18: end procedure
```

3.2.3.4 Šenono entropija

Sekų suderinamumui skaičiuoti taip pat bus pasitelkiamas Šenono entropijos skaičiavimas. Paprastos entropijos skaičiavimas parodo duomenų rinkinio atsitiktinumą/netvarkingumą. Šenono entropija nurodo įsitikinimo reikšmę - kuo reikšmė yra didesnė, tuo didesnis įsitikinimas turint atitinkamus duomenis gauti tikėtiną rezultatą [Acz06]. Skaičiavimas yra pateiktas formulėje 7 [Acz06]. Entropijos reikšmė yra žymima raide H. E yra duomenų skirtinio funkcija. I yra kintamojo X turinys.

$$H(X) = E[I(X)] = E(-\log_p(X)) \tag{7}$$

Entropijos funkcija taip pat gali būti išreikšta formulėje 8 [Acz06]. Kintamasis b - kintanti logaritmo bazė. Reikšmė paprastai būna viena iš: 2, e arba 10. Funkcija $p(x)$ - tikimybės funkcija. Šios



14 pav. Sekų aibės konservatyvumo reikšmės

tikimybės deterministiškai neįmanoma apibrėžti, todėl funkcijos sukūrimui bus naudojami eksperimentiniai duomenys. Surinkti duomenų rinkiniai ir gauti rezultatai surašomi į duomenų skirstinį. Bus bandoma sukurti skirstiniu grįstą sekų hidrofobiškumo skaičiavimą. Detalus algoritmas pateiktas diagramoje 4.

$$H(X) = \sum_{x \in X} p(x) \log_b p(x) \tag{8}$$

Šis metodas leis sukurti preliminarų duomenų rinkinį ir gauti jo įsitikinimo reikšmę. Taikant šią transformacijos funkciją duomenų aibei, o ne pavienėms sekoms, tikėtini geresni rezultatai.

3.2.4 Rezultatai

Tyrime buvo realizuotos minėtos funkcijos ir sukurti keli mašininio mokymosi modeliai funkcijų reikšmingumui patikrinti. Šiame bandyme modeliai analizavo visus transformacijų funkcijų duomenis vienu metu. T.y. kiekvieno modelio įvestis buvo visi gauti duomenys iš minėtų funkcijų. Šis algoritmas yra pateiktas 2 algoritmo schemoje. Gauti rezultatai pateikti lentelėje 6. Sukurtas neuroninis tinklas naudoja sigmoidinę aktyvacijos funkciją, kuri pateikta formulėje 9. Mokymosi funkcija (angliškai „Solver“), kuri yra naudojama modelio optimizavimui, yra L-BFGS-B. T.y. modifikuotas Broyden–Fletcher–Goldfarb–Shanno algoritmas. Sukurtas AdaBoost modelis yra tiksliausias iš visų sukurtų. Modelio kūrimui pasirinkta naudoti SAMME.R klasifikavimo algoritmo kvadratinę nuostolių funkciją, gradiento didinimui.

Algoritmas 4 Šenono algoritmas

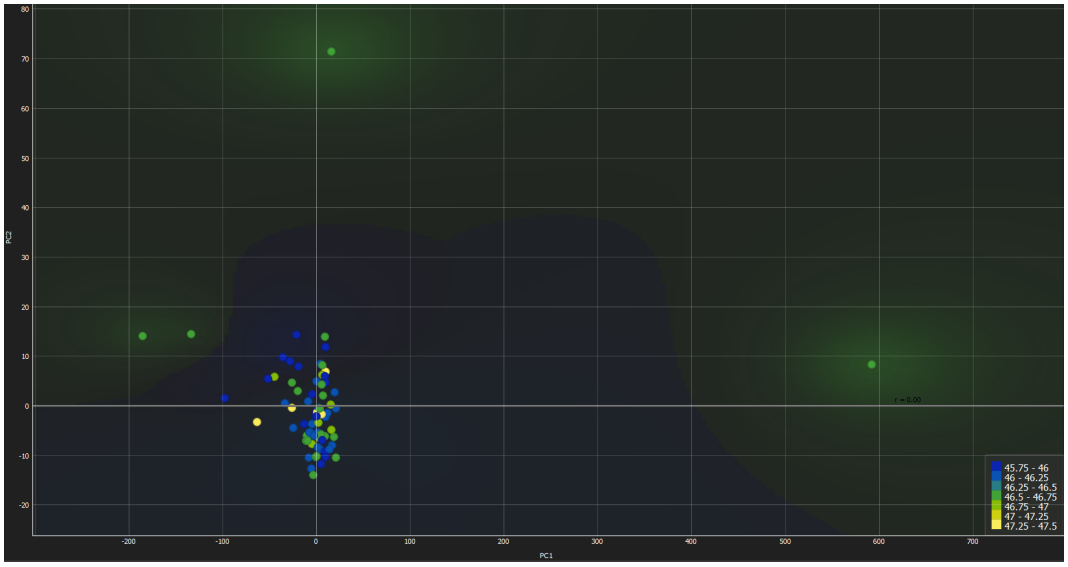
```
1: procedure ŠenonoEntropija(msa)
2:     n ← Sekų skaičius MSA struktūroje
3:     m ← Sekų ilgis
4:     e ← Tuščias masyvas, skirtas saugoti entropijos reikšmes
5:     for i in range(n) do
6:         aminoRugstys ← SurastiAminoRugstis(msa, i)
7:         tikimybes ← TikimybiuRadimas(aminoRugstys, m)
8:         entropija ← RastiEntropija(tikimybes)
9:         e.push(entropija)
10:    end for
11:    return e
12: end procedure
13: procedure SurastiAminoRugstis(msa, i)
14:    rugstys ← Tuščia HashMap struktūra
15:    for seka in msa do
16:        rugstys[seka[i]] ← rugstys.get(seka[i], 0) + 1
17:    end for
18:    return rugstys
19: end procedure
20: procedure TikimybiuRadimas(aminoRugstys, n)
21:    tikimybes ← Tuščias masyvas
22:    for rugstis, i in aminoRugstys.items() do
23:        tikimybe ← i/n
24:        tikimybes[rugstis] ← tikimybe
25:    end for
26:    return tikimybes
27: end procedure
28: procedure RastiEntropija(tikimybes)
29:    e ← 0
30:    for tikimybe in tikimybes.values() do
31:        entropija ← e − tikimybe · log2(tikimybe)
32:    end for
33:    return e
34: end procedure
```

Modelis	MSE	RMSE	MAE	R2
Neuroninis tinklas	9,128	3,021	2,050	-39,284
Tiesinė regresija	11,711	3,422	1,481	-50,685
Gradient boosting	0,379	0,616	0,505	-0,673
AdaBoost	0,355	0,596	0,4666	-0,568

6 lentelė. Modeliai

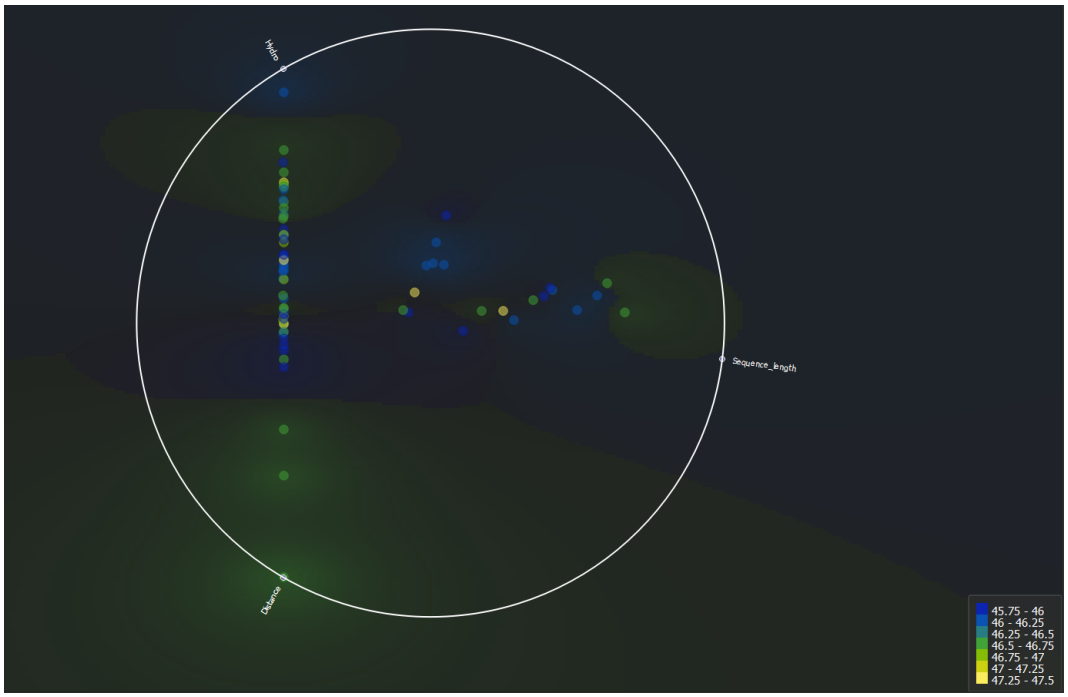
$$f(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

Taikant PCA metodą, nepavyko rasti reikšmingų koreliacijų. Su 2 PCA komponentėmis pavyko pasiekti 99% parametrų padengiamumą, tačiau vizualiai jokios koreliacijos nesimato. PCA vizualizacija pateikta pav. 15.



15 pav. PCA komponentės

Koreliacijų buvo taip pat bandoma ieškoti taikant Radviz vizualizavimą. Jokių koreliacijų nepavyko rasti. Rezultatai pateikti pav. 16.



16 pav. Radviz

Koreliacijoms ieškoti taip pat buvo skaičiuojamos P reikšmės, pasinaudojant Spirmano ir Pirsono metrikomis. Rezultatai pateikti lentelėse 7 ir 8. Pagal Pirsono metriką visi minėti rodikliai yra statistiškai reikšmingi. Pagal Spirmano metriką vienintelis sekos ilgis yra statistiškai nereikšmingas. Pagal PCA analizę ir koreliacijas tiesinės funkcijos parametrų reikšmingumui nepavyko apibrėžti, todėl tyrimuose bus naudojamas tiksliausias sukurtas modelis - AdaBoost.

P reikšmė	Kintamasis	Reikšmingumas
-0,083	Sekos ilgis	Statistiškai nereikšminga
0,032	Hidrofobiškumas	Statistiškai reikšminga
-0,030	Konservatyvumas	Statistiškai reikšminga

7 lentelė. Spirmano metrika

P reikšmė	Kintamasis	Reikšmingumas
-0,037	Sekos ilgis	Statistiškai reikšminga
0,027	Konservatyvumas	Statistiškai reikšminga
0,019	Hidrofobiškumas	Statistiškai reikšminga

8 lentelė. Pirsono metrika

Pritaikius pasiūlytą sprendimą DIAMOND ir SWIPE atrinktiems duomenims ir sujungus juos su MMSeq2 algoritmo atrinktais duomenų rinkiniais, išvesta kokybės metrika išlieka ta pati, jokio rezultatų pagerėjimo nėra. Atrinktų duomenų rinkinio mažinimo strategija nepasitvirtino.

4 Tolimesni galimi patobulinimai

4.1 Siūlomas skaičiavimo spartinimas

SWIPE algoritmas naudoja procesoriaus gijas paprastų skaičiavimų paralelizavimui. Tai nėra pats efektyviausias metodas paralelizuoti paprastas operacijas. Procesoriai įprastai turi nuo 4 iki 8 branduolių. Vaizdo plokštės turi tūkstančius branduolių. Jie yra silpnesni už procesoriaus branduolius, tačiau, jeigu viena gija dirba su viena ar tik keliomis atominėmis operacijomis, o programa efektyviai manipuliuoja plokštės atmintimi, gijų greitaveika yra panaši. Atominės operacijos, kaip sudėtis ar atimtis gali būti efektyviai paralelizuotos, pasitelkus vaizdo plokštės gijas. Vaizdo plokštės gali sukurti dešimtis tūkstančių gijų vienu metu, o procesorius tik kelias dešimtis [NVF20].

Šiame skyriuje bus plačiau nagrinėjama Smith-Waterman algoritmo versija. Lyginant dvi sekas, iš kurių pirmos ilgis yra N_1, antros N_2, algoritmo kompleksiškumas yra $O(3N_1N_2)$. Taikant paralelizavimą, kompleksiškumą galima sumažinti iki $O(3(N_1 + N_2 - 1))$. Lentelėje 9 pateiktas reikalingas operacijų skaičius, kuris priklauso nuo sekų ilgis.

Tarkime, kad turime dvi sekas „TGAC“, „GAC“ ir bandome jas sulygiuoti. Tam reikia apskaičiuoti poslinkių reikšmių matricą. Tarkime, kad baudos neatitikimo reikšmė -1, tarpo -2, atitikimo 1. Viena seka yra išdėstoma paraleliai, kita horizontaliai ir skaičiuojama poslinkių matrica. Pradinė poslinkių matrica:

$$\begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & & & & \\ A & -4 & & & & \\ C & -6 & & & & \end{bmatrix}$$

Toliau skaičiavimai yra vykdomi tikrinant viršutinę, kairę ir įstrižą reikšmes. Horizontalus tikrinimas yra sekos poslinkis, įstrižas - elementų lyginimas tarpusavyje. Kiekviena sekanti matrica atvaizduoja skaičiavimų iteraciją. Kiekvienas paryškintas elementas yra skaičiuojamas atskiros gijos.

$$\begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & \textbf{-1} & & & \\ A & -4 & & & & \\ C & -6 & & & & \end{bmatrix} \begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & -1 & \textbf{-1} & & \\ A & -4 & \textbf{-3} & & & \\ C & -6 & & & & \end{bmatrix} \begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & -1 & -1 & \textbf{-3} & \\ A & -4 & -3 & \textbf{-2} & & \\ C & -6 & \textbf{-5} & & & \end{bmatrix}$$

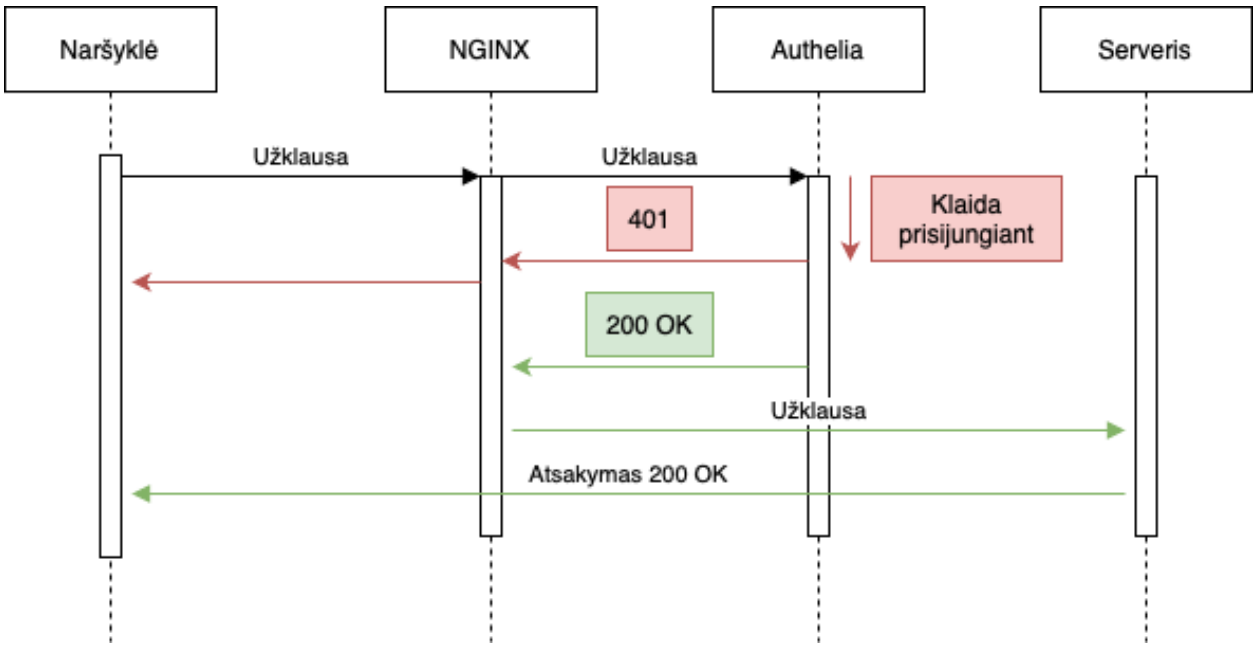
$$\begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & -1 & -1 & -3 & \textbf{-5} \\ A & -4 & -3 & -2 & \textbf{0} & \\ C & -6 & -5 & \textbf{-4} & & \end{bmatrix} \begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & -1 & -1 & -3 & -5 \\ A & -4 & -3 & -2 & 0 & \textbf{-2} \\ C & -6 & -5 & -4 & \textbf{-3} & \end{bmatrix} \begin{bmatrix} - & 0 & T & G & A & C \\ 0 & 0 & -2 & -4 & -6 & -8 \\ G & -2 & -1 & -1 & -3 & -5 \\ A & -4 & -3 & -2 & 0 & -2 \\ C & -6 & -5 & -4 & -3 & \textbf{1} \end{bmatrix}$$

Kiekvienos sekos ilgis	1	2	3	4	5	6	7	8	9
Operacijų kiekis neparalelizavus	3	12	27	48	75	108	147	192	243
Operacijų kiekis paralelizavus	3	9	15	21	27	33	39	45	51
Pagreitėjimas	0%	25%	44%	56%	64%	69%	73%	77%	79%

9 lentelė. Algoritmo paralelizavimo rezultatai

Didžiausias panaudojamų gijų skaičius yra lygus trumpiausios sekos ilgiui. Paralelizavimo našumas pasireiškia lyginant ilgesnes sekas tarpusavyje.

Algoritmas turi atminties apribojimus. Naudojant paprastą procesorių yra kuriamos gijos ir per daug nesirūpinama atminties naudojimu. Naudojant grafikos plokštes, reikia daug dėmesio kreipti į naudojamas atminties struktūras. Grafikos plokštės įprastai turi dviejų tipų atmintis - „DRAM“ globali atmintis, kuri yra naudojama numatytu atveju ir „Cache“ atmintis, kuri yra 2 kartus greitesnė už „DRAM“ atmintį. Skirtingai nuo procesoriaus, grafikos plokštės neturi vieno „Cache“ komponento, jų yra daug. NVIDIA vaizdo plokštės savo branduolius yra išdėsčiusios N*M struktūroje, kurioje kiekviena eilė branduolių turi savo atskirą „Cache“ komponentą. Šią atskirą branduolių eilę su savo atskiru atminties komponentu NVIDIA vadina bloku. Tai reiškia, kad pasiūlytame paralelizuotame algoritme yra atitinkamas sekų ilgių limitas, priklausomai nuo turimos vaizdo plokštės. Riba priklauso nuo branduolių skaičiaus vaizdo plokštės blokuose. Prireikus daugiau branduolių, nei yra išdėstyta bloke, reiktų arba vykdyti duomenų sinchronizaciją tarp kiekvieno bloko, arba naudoti globalią „DRAM“ atmintį, kuri yra 2 kartus lėtesnė [NVF20].



17 pav. Tarpininkavimo schema

4.2 Duomenų saugumas

Sukurtas sprendimas nėra pilnai saugus. Nėra jokių įgyvendintų metodų, skirtų apsisaugoti nuo kenkėjiškų vartotojų, bandančių atspėti prisijungimo raktą. Vienas iš galimų sprendimų yra naudoti NGINX srauto ribojimo funkcionalumą. Jis naudoja kiauro kibiro algoritmą. Užklausos, kurios pirmos patenka į sąrašą, yra pirmiau aptarnaujamos (naudojama „FIFO“ sistema). Jei sąrašas persipildo, užklausos bandančios patekti į sąrašą, yra išmetamos. Sąrašo apdorojimo greitį leidžia nustatyti NGINX programinė įranga. Sąrašai, į kuriuos pakliūna užklausos, gali būti keičiami - pagal IP adresą, užklausos adresą ar kitus parametrus. Šiuo atveju reikėtų naudoti užklausų valdymą pagal IP adresą. Tai leistų vartotojams laisvai naudotis programine įranga, tačiau atsiradus užklausų pertekliui iš atitinkamo IP adreso, užklausos, kurios sudaro perviršį, yra išmetamos [DeJ20].

Autentifikavimo ir autorizavimo logiką taip pat galima iškelti į NGINX. Egzistuoja tapatybių valdymo sistemos, tokios kaip Authelia, kurio veikia kaip atvirkštinis tarpininkas („reverse-proxy“). Šias sistemas galima sukonfigūruoti taip, kad veiktų kartu su NGINX [DeJ20]. Vartotojų užklausos pradžioje keliautų į NGINX programinę įrangą. Joje turi būti sukonfigūruotas tarpininkavimo sluoksnis, kad užklausas patikrintų Authelia tapatybių valdymo sistema. Jei užklausa yra patvirtinta, NGINX pateikia užklausą į serverį. Taip būtų galima atsisakyti sukurtos autentifikavimo sistemos, kuri neturi jokių papildomų saugumo mechanizmų. Veiksmų schema yra pateikta diagramoje 17.

Rezultatai

1. Darbe buvo praplėsta ColabFold programa, leidžiant nurodyti savo pradinių duomenų paieškos serverio adresą. Taip yra leidžiama modifikuoti esamą integraciją su MMSeq2 sprendimu.

Praplėtimas leidžia nurodyti savo pradinių duomenų paieškos serverio adresą. Pateiktas pradinių duomenų paieškos serverio sprendimas, kuris leidžia naudoti kelis paieškos algoritmus, bei dinamiškai pridėti duomenų bazes programos veikimo metu. Dabartinis sprendimas, kurį naudoja ColabFold, leidžia naudoti vienintelį MMseq2 sekų paieškos algoritmą, o duomenų bazės gali būti pridedamos tik prieš programos paleidimą.

2. ColabFold praplėstas DIAMOND ir SWIPE programomis.

Buvo įvykdyti 2 bandymai. Seką, kuri turi kokybišką susilankstymo kokybės metriką pLLDT 97,4, pavyko pagerinti pasirinkus MMSeq2 praplėtimą su DIAMOND programa. Kokybės metrika pagerėjo 0.01%. Pasirinkus seką, kurios ColabFold su MMSeq2 negali kokybiškai sulankstyti - pLDDT 43,3, pavyko pagerinti pasirinkus MMSeq2 duomenų praplėtimą su SWIPE programa. Rezultatai pagerėjo 29,6% - pLDDT 56,1. Nors bendra pLDDT reikšme yra maža, pažvelgus į susilankstymo rezultatus pateiktus pav. 10 matosi, kad didelė dalis baltymo turi neblogą pLDDT reikšmę, o likusi dalis prastą reikšmę.

3. Bandymas skaičiuoti suderinamumo indekso reikšmes sekoms buvo nesėkmingas.

Atrinkus tik dalį sekų, kuriomis turėjo būti papildytas MMSeq2 atrinktas duomenų rinkinys, rezultatai išlikdavo tokie patys. Visais atvejais rezultatai yra geresni palikus pilnus atrinktus duomenų rinkinius. Bandyta taip pat šį indeksavimą naudoti naujai pridėtiems duomenims iš SWIPE ir DIAMOND programų, tačiau tai taip pat nebuvo sėkmingas bandymas. Mažinant duomenų aibę, gautą iš papildomų programų, rezultatai įprastai būdavo prastesni, nei su pilnu duomenų rinkiniu.

Išvados

1. Pasiūlyta metrika skaičiuoti suderinamumo indeksą nėra veiksminga.

Ši metrika geriausiu atveju gali padaryti programos veikimą nedeterministišku, tačiau rezultatai nebus geresni. Pateikus keletą metodų baltymų lyginimui ir sukūrus mašininio mokymosi modelių ansamblių architektūrą, nepavyko pasiekti nedeterministiškumo. Indeksavimo taip pat negalima pritaikyti pridėtinių duomenų analizei.

2. Sekų paieškos serveris, kurį naudoja ColabFold, yra per daug apribotas.

MMSeq2 serveris neleidžia pridėti savo sekų paieškos programų, bei neleidžia programos vykdymo metu dinamiškai pridėti reikalingas duomenų bazes. Ši problema išspręsta - darbe pateiktas sprendimas, kuriame galima lengvai pridėti savo sekų paieškos programas, pridedant naują Dockerfile. Sekų duomenų bazes taip pat galima pridėti programos vykdymo metu.

3. ColabFold per daug riboja pradinio duomenų rinkinio sudarymo nustatymus.

Darbe pateikta modifikacija išsprendžia ColabFold ribotumą ir leidžia nurodyti savo pradinių duomenų paieškos sprendimo IP adresą, naudojamą duomenų bazę, metodą ir duomenų kiekį.

Literatūros sąrašas

[Acz06] János Aczél. Entropies, characterizations, applications and some history. In Bernadette Bouchon-Meunier, Giulianella Coletti, and Ronald R. Yager, editors, *Modern Information Processing*, pages 3–10. Elsevier Science, Amsterdam, 2006.

[BWF+00] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 01 2000.

[Car23] Oliviero Carugo. Plddt values in alphafold2 protein models are unrelated to globular protein local flexibility. *Crystals*, 13(11):1560, Nov 2023.

[CCD18] Mary Ann Clark, Jung Choi, and Matthew Douglas. *12 Proteins*. OpenStax, 2018.

[Dar21] Steve Darnell. Why structure prediction matters, May 2021.

[DeJ20] D. DeJonghe. *NGINX Cookbook*. O'Reilly Media, 2020.

[HSS16] Maria Hauser, Martin Steinegger, and Johannes Söding. MMseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics*, 32(9):1323–1330, 01 2016.

[IG07] Zoya Ignatova and Lila M. Gierasch. Protein folding, 2007.

[JEP+21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, and et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, Jul 2021.

[LM21] Michael J. Lopez and Shamim S. Mohiuddin. *Biochemistry, Essential Amino Acids*. 2021.

[M10] O'Connor C. M. *How Do Cells Decode Genetic Information into Functional Proteins?* NPG Education, 2010.

[MSM+22] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: Making protein folding accessible to all. *Nature Methods*, 19(6):679–682, May 2022.

[NVF20] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[Pon91] P. K. Ponnuswamy. Hydrophobic characteristics of folded proteins. *Proteins*, page 361–366, 1991.

[Rab23] RabbitMQ Team. Rabbitmq documentation, 2023. Accessed: February 16, 2024.

[Red08] Michael K. Reddy. Amino acid, May 2008.

[Rog11] Torbjørn Rognes. Faster smith-waterman database searches with inter-sequence simd parallelisation. *BMC Bioinformatics*, 12(1), Jun 2011.

[Ryu17] Wang-Shick Ryu. X-ray crystallography, 2017.

[SFB04] Peter D. Sun, Christine E. Foster, and Jeffrey C. Boyington. Overview of protein structural and functional folds. *Current Protocols in Protein Science*, 35(1), 2004.

[SS17] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, Oct 2017.

[YCZW10] Lianping Yang, Guisong Chang, Xiangde Zhang, and Tianming Wang. Use of the burrows–wheeler similarity distribution to the comparison of the proteins. *Amino Acids*, 39(3):887–898, Mar 2010.

Priedas Nr. 1

1. ColabFold praplėtimas - https://github.com/PauliusMilmantas/ColabFoldWebServer
2. Suderinamumo indekso skaičiavimai - https://github.com/PauliusMilmantas/BaigiamasisDarbas