UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ALGORITMUS PRE HĽADANIE VZDIALENÝCH FUNKČNÝCH ORTOLÓGOV A JEHO APLIKÁCIA NA OB-FOLD DOMÉNU

Dizertačná práca

2012
Mgr. Martin Macko

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ALGORITMUS PRE HĽADANIE VZDIALENÝCH FUNKČNÝCH ORTOLÓGOV A JEHO APLIKÁCIA NA OB-FOLD DOMÉNU

Dizertačná práca

Študijný program: Informatika

Školiace pracovisko: Katedra Aplikovanej Informatiky

Školiteľ: Mgr. Tomáš Vinař, PhD.

Bratislava 2012
Mgr. Martin Macko

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Mgr. Martin Macko
Študijný program: informatika (Jednoodborové štúdium, doktorandské III. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: dizertačná
Jazyk záverečnej práce: slovenský

Názov: Algoritmus pre hľadanie vzdialených funkčných ortológov a jeho aplikácia na OB-fold doménu

Cieľ: Cieľom práce je vytvoriť nový algoritmus pre hľadanie vzdialených funkčných ortológov, ktorý umožní využitie expertných doménových znalostí používateľa. Implementáciu algoritmu je potrebné otestovať na skutočných dátach, pričom vhodnou aplikáciou je hľadanie proteínov s OB-fold doménou.

Školiteľ: Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky

Dátum zadania: 01.09.2011

Dátum schválenia: 16.07.2012 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
školiteľ

Poďakovanie

Ďakujem môjmu školiteľovi Mgr. Tomášovi Vinařovi PhD. a Mgr. Broni Brejovej PhD. za trpezlivosť, vedenie a cenné rady pri písaní tejto práce ako aj počas celého doktorandského štúdia. Ďalej sa chcem poďakovať Mgr. Martinovi Králikovi za spoluprácu na materiály obsiahnutom v kapitole 6. V neposlednom rade sa chcem poďakovať mojej rodine a priateľom za podporu počas môjho štúdia.

Abstract

Identification of evolutionarily related proteins (orthologs) is an important step towards understanding protein function in living organisms. This problem is difficult in many cases because distantly related proteins have often too divergent amino-acid sequences, so this relation may not be identified by traditional methods based mainly on sequence similarity. In these cases search should use combination of structural and sequential features of protein.
In this thesis, we propose a new approach to the search for remote functional orthologs, where important features of protein sequence and structure are represented by a descriptor created by human expert. This descriptor can be used to search for described features in candidate protein sequences. For the problem of ortholog identification with descriptor, we develop scoring scheme which combines sequence profiles and support vector machines to evaluate alignment of the descriptor to the candidate sequence, and we develop algorithms which that the best possible alignment.
We demonstrate our approach on the example of telomere-binding OB-fold domain. Our method can distinguish between Telo_bind family members and negatives, and also identifies proteins containing a related OB-fold domain.

Keywords: protein ortholog identification, support vector machines, integer linear programming, dynamic programming

Abstrakt

Identifikácia evolučne príbuzných proteínov (ortológov) je dôležitým krokom k pochopeniu funkcie proteínov v živých organizmoch. Tento problém je v mnohých prípadoch obtiažny, pretože vzdialene príbuzné proteíny majú často príliš rozdielne sekvencie aminokyselín, takže tento vzťah nemusí byť identifikovateľný tradičnými metódami založenými na sekvenčnej podobnosti. Pre tieto prípady je potrebné použiť pri hľadaní kombináciu vlastností sekvencie a štruktúry proteínu.
V tejto práci navrhujeme nový prístup k hľadaniu vzdialených funkčných ortológov, v ktorom sú dôležité vlastnosti sekvencie a štruktúry domény hľadaného proteínu reprezentované ručne zostaveným deskriptorom. Tento deskriptor sa dá použiť na hľadanie popísaných vlastností v kandidátskych sekvenciách. Pre problém hľadania ortológov pomocou deskriptora sme vyvinuli skórovaciu schému, ktorá kombinuje sekvenčné profily a metódu support vector machines na ohodnotenie zarovnania deskriptora ku kandidátskej sekvencii. Ďalej sme vyvinuli algoritmy, ktoré nájdu najlepšie takéto zarovnanie.
Náš prístup sme demonštrovali na príklade OB-fold domény viažúcej teloméry. Naša metóda dokáže rozlíšiť členov rodiny Telo_bind a negatívne príklady a navyše identifikuje aj proteíny obsahujúce príbuznú OB-fold doménu.

Kľúčové slová: identifikácia ortologických proteínov, support vector machines, celočíselné lineárne programovanie, dynamické programovanie

Obsah

Úvod

Základom fungovania živých organizmov je využívanie genetickej informácie uloženej v chromozómoch na tvorbu rozličných proteínov. Biologická funkcia proteínu v bunke závisí nielen od sekvencie aminokyselín, z ktorej sa skladá, ale aj od tvaru štruktúry, ktorú zaujme v trojrozmernom priestore. Na odhalenie týchto funkčných závislostí medzi biologickými procesmi a genetickými informáciami, je kvôli objemu dát nutné použiť prostriedky informatiky. Niektoré proteíny napriek tomu, že vystupujú v esenciálnych biologických procesoch, nemajú v rámci evolúcie výrazne zachovanú informáciu o biologickej funkcii v sekvencii aminokyselín, ale v 3D štruktúre, čo výrazne sťažuje vyhľadávanie týchto proteínov pomocou konvenčných metód založených na sekvenčnej podobnosti proteínov.

Podobný problém sa vyskytuje aj pri hľadaní RNA génov, kde je sekundárna štruktúra RNA kľúčová pre identifikovanie evolučne príbuzných génov. Okrem plne automatických metód sa pri riešení tohto problému osvedčili nástroje, ktoré umožňujú odborníkom ručne zostaviť špecializované deskriptory, ktoré charakterizujú najdôležitejšie vlastnosti hľadanej sekvencie vybrané expertmi. Tieto deskriptory popisujú vybrané črty pomocou sady obmedzení, ktoré by mala spĺňať sekvencia s hľadanými vlastnosťami. V tejto práci navrhneme rozšírenie tohto prístupu pre identifikovanie vzdialených evolučne príbuzných proteínov. Charakteristické vlastnosti hľadanej sekvencie sa však pre proteíny nedajú vyjadriť jednoduchými obmedzeniami ako tomu je v prípade RNA génov. Na reprezentáciu týchto dôležitých vlastností hľadanej proteínovej sekvencie, budeme v našej práci kombinovať metódy strojového učenia (support vector machines), pravdepodobnostného modelovania (sekvenčné profily) spolu s ručne vybranými vlastnosťami štruktúry popisovanej

sekvencie.

V tejto práci popíšeme metódy, ktoré sa zaoberajú hľadaním konkrétnych génov v evolučne príbuzných organizmoch, rozoberieme najčastejšie používané nástroje pri tejto analýze a rôzne prístupy, ktoré sa používajú na zlepšenie senzitivity tohto hľadania. Tieto metódy sú úspešné pre veľké množstvo génov, ale nie vždy nájdu gény pre evolučne vzdialenejšie proteíny.

Preto cieľom tejto práce je vytvoriť deskriptor štrukturálnej domény hľadaného proteínu (úsek sekvencie so špecifickými vlastnosťami) na základe zachovaných spoločných prvkov v sekvencii a štruktúre evolučne príbuzných proteínov. Ďalej je potrebné vyvinúť efektívny algoritmus pre identifikáciu proteínov s doménou popísanou týmto deskriptorom. Takýto prístup by mohol rozšíriť senzitivitu hľadania na proteíny, ktoré sa sekvenciou výraznejšie odlišujú od známych proteínov s danou štrukturálnou doménou.

Prvá kapitola sa zameriava na predstavenie základných pojmov v oblasti analýzy biologických sekvencií, evolučných vzťahov a vyhľadávania príbuzných proteínov. Druhá kapitola obsahuje prehľad jednotlivých metód používaných na identifikáciu evolučne príbuzných génov v rôznych organizmoch a popisuje princípy, na ktorých sa tieto metódy zakladajú.

Tretia kapitola popisuje špecializovaný sekvenčný profil (deskriptor) pre štrukturálne domény, ktorý kombinuje informácie o sekvencii a štruktúre popisovanej domény. V tejto kapitole taktiež definujeme problém zarovnania tohto deskriptoru k sekvencii a skórovaciu schému pre ohodnotenie takéhoto zarovnania. Kapitola zároveň obsahuje výsledky analýz, ktoré boli urobené existujúcimi metódami na génoch pre proteíny telomerického komplexu v rôznych kvasinkových druhoch. Na základe týchto výsledkov špecifikuje deskriptor pre štrukturálnu doménu Telo_bind, ktorá viaže telomerické proteíny na DNA sekvenciu na koncoch chromozómov.

Štvrtá kapitola sa zaoberá vývorom spôsobu ohodnocovania potenciálnych interakcií medzi vzdialenými pozíciami v sekvencii proteínu. Tieto interakcie sú jednou z vlastností hľadanej domény popísaných vo vytvorenom deskriptore a špecifikujú štruktúru, do ktorej sa sekvencia proteínu skladá. Na identifikovanie interagujúcich pozícií sme využili metódu strojového učenia support vector machines, pomocou ktorej sme natrénovali na reálnych

dátach niekoľko rôznych klasifikátorov. Štvrtá kapitola obsahuje popis metódy support vector machines, použitie tejto metódy na vytvorenie skupiny klasifikátorov a analýzu výsledkov pre testovanie natrénovaných klasifikátorov.

Piata kapitola popisuje riešenie problému určenia najlepšieho zarovnania deskriptora k sekvencii proteínu pomocou celočíselného lineárneho programovania a obsahuje výsledky experimentov pre tento spôsob riešenia problému zarovnania. Tento prístup sa ukázal nepraktický pre použitie na veľké množstvo sekvencií, a preto v poslednej kapitole predstavíme relaxovaný problém zarovnania pre špecifickú podtriedu deskriptorov. Šiesta kapitola sa zaoberá zjednodušeným podproblémom nájdenia najlepšieho zarovnania pre špecifické deskriptory, ktorý je možné riešiť dynamickým programovaním v polynomiálnom čase. Zároveň sú v tejto kapitole popísané výsledky experimentov s touto metódou na vybranej skupine proteínov ako aj na genóme kvasinky *Yarrowia lipolytica*, ktoré dokumentujú aplikovateľnosť nami vytvorenej metódy na identifikovanie vzdialených funkčných ortológov na reálnych biologických dátach.

Kapitola 1

Problém hľadania ortologických komplexov

Spracovať údaje o geneteckých informáciách rôznych organizmoch, ktoré sú výsledkom biologických experimentov je kvôli ich objemu veľmi náročné a preto je vhodné použiť na ich analýzu informatické metódy. V tejto kapitole popíšeme základné biologické pojmy pre genetické informácie uložené v organizmoch ako a aj evolučné procesy, ktoré túto informáciu využívajú. Na základe tejto kapitoly v ďalších kapitolách formulujeme informatický problém, ktorým sa v tejto práci zaoberáme.

1.1 Bunka, genóm, gén, proteín

Pretože mnohé vlastnosti buniek ostávajú zachované v priebehu evolúcie, poznatky o biologických procesoch v jednobunkových organizmoch (napríklad kvasinkách) sa často dajú aplikovať na bunky vyšších organizmov, čo sa využíva v medicíne či iných oblastiach. Navyše genetické sekvencie kvasiniek sú mnohonásobne menšie než genómy vyšších organizmov, preto sú vhodnejšie na analýzu výpočtovo náročnými algoritmami (Botstein et al., 1997).

Kompletná genetická informácia organizmu sa nazýva *genóm*. *Gén* je úsek genetickej sekvencie zodpovedajúci niektorej funkčnej jednotke (proteín, RNA gén a podobne). Z génov sa v bunke syntetizujú proteíny procesmi, ktoré vedú od genetickej informácie uloženej v molekulách DNA (deoxyribonukleová kyselina), cez molekuly RNA (ribonuk-

leová kyselina) k proteínom.

DNA sú dlhé molekuly pozostávajúce z pospájaných purínových (adenín, guanín) a pyrimidínových báz (cytozín, tymín). DNA molekuly sú dvojvláknové a bázy na jednotlivých vláknach sa komplementárne dopĺňajú. Cytozín sa viaže s guanínom a adenín s tymínom (C-G, A-T). Tieto nukleotidové bázy (4 písmenová abeceda) predstavujú prvú úroveň genetického kódu. RNA má na rozdiel od DNA len jedno vlákno a namiesto nukleotidovej bázy tymín obsahuje uracil (U).

Pri procese syntézy proteínu sa najprv z molekuly DNA prepíše genetická informácia do molekuly RNA. Tento proces sa nazýva *transkripcia*. Informácia uložená v takto vytvorenej RNA molekule je následne preložená do sekvencie aminokyselín - proces *translácie*. Výsledná sekvencia aminokyselín sa poskladá do trojrozmernej štruktúry a vytvorí proteín, ktorý vykonáva v bunke potrebnú biologickú funkciu. Schéma týhto procesov je na obrázku 1.1.

Trojica nukleotidových báz sa nazýva *kodón*. Kodóny sa prekladajú (proces *translácie*) na aminokyseliny, ktoré tvoria druhú úroveň genetického kódu (20 písmenová abeceda). Niektoré aminokyseliny kóduje viac rôznych kodónov. Sekvencia vždy začína iniciačným kodónom (aminokyselina metionín). Špeciálne kodóny označujú tiež koniec kódujúcej sekvencie.

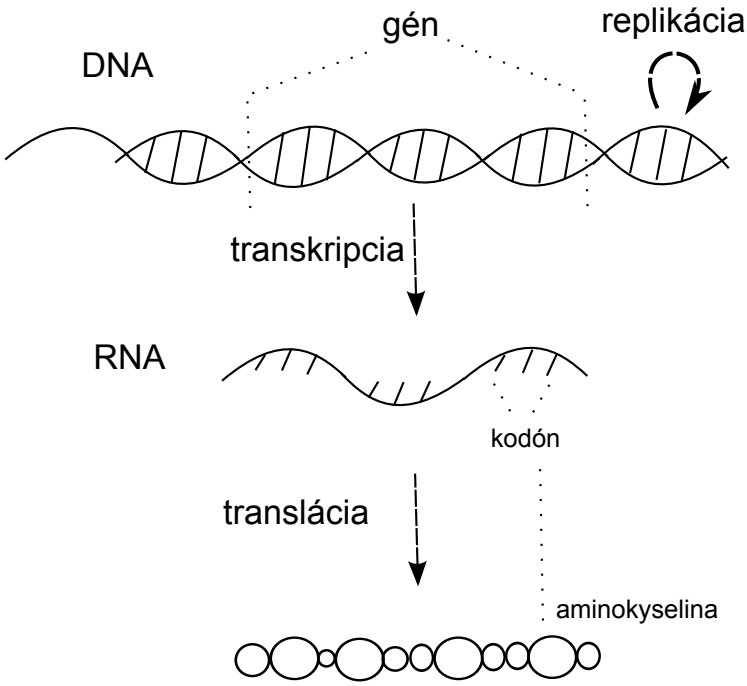
V každej bunke je veľké množstvo rozličných proteínov, ktoré vystupujú v rôznych funkciách v biologických procesoch. Proteíny sú základnými stavebnými časťami buniek, katalyzujú reakcie, pôsobia ako protilátky a majú mnohé ďalšie funkcie. Funkcia proteínu závisí od chemických vlastností aminokyselín, z ktorých sa skladá, ako aj od trojrozmernej štruktúry, ktorú nadobudne poskladaný reťazec aminokyselín v priestore.

1.2 Štruktúra proteínov a proteínové domény

Primárna štruktúra proteínu predstavuje aminokyseliny, z ktorých sa skladá. Pod *sekundárnou štruktúrou* rozumieme popísanie tejto sekvencie z hľadiska lokálnych vodíkových väzieb medzi susediacimi aminokyselinami. Vďaka týmto väzbám nadobúdajú časti sekven-

Aminokyselina	Kódovaná písmenom	DNA kodón
Alanín	A	GCT, GCC, GCA, GCG
Cysteín	C	TGT, TGC
Kyselina asparágová	D	GAT, GAC
Kyselina glutamátová	E	GAA, GAG
Fenylalanín	F	TTT, TTC
Glycín	G	GGT, GGC, GGA, GGG
Histidín	H	CAT, CAC
Izoleucín	I	ATT, ATC, AUA
Lyzín	K	AAA, AAG
Leucín	L	TTA, TTG, CTT, CTC, CTA, CTG
Metionín	M	ATG
Asparagín	N	AAT, AAC
Prolín	P	CCT, CCC, CCA, CCG
Glutamín	Q	CAA, CAG
Arginín	R	CGT, CGC, CGA, CGG, AGA, AGG
Serín	S	TCT, TCC, TCA, TCG, AGT, AGC
Threonín	T	ACT, ACC, ACA, ACG
Valín	V	GTT, GTC, GTA, GTG
Tryptofán	W	TGG
Tyrozín	Y	TAT, TAC
Stop kodóny	Stop	TAA, TAG, TGA

Tabuľka 1.1: Zoznam aminokyselín, ich označenia a DNA kodónov, ktoré ich kódujú.



Obr. 1.1: Základné biologické procesy, v ktorých vystupuje genetická informácia.

cie pravidelný tvar (závitnica α-helix alebo vyrovnaný β-list). Identifikovanie sekundárnej štruktúry proteínu spočíva v označení takýchto úsekov v primárnej štruktúre. Sekvencia aminokyselín sa navyše poskladá do konkrétneho tvaru, v ktorom proteín plní svoju funkciu. Pozície jednotlivých atómov v aminokyselinách proteínu v trojrozmernom priestore určujú *terciárnu štruktúru*. Náčrt postupnosti proteínových štruktúr je zobrazený na obrázku 1.2. Zistiť trojdimenzionálnu štruktúru proteínu je experimentálne náročné (kryštalografia, magnetická rezonancia), a preto boli vyvinuté výpočtové metódy na predikciu sekundárnej a terciárnej štruktúry, ktoré umožňujú pre danú vstupnú sekvenciu aminokyselín odhadnúť jej štruktúru (Jones, 1999; Rost et al., 1994; Zimmermann, 2003).

Doména proteínu je úsek sekvencie aminokyselín, ktorý zodpovedá za určitú chemickú vlastnosť proteínu alebo funkciu, ktorú proteín vykonáva (napríklad viazať, fyzicky interagovať s inými proteínmi alebo viazať sa na jednovláknovú molekulu DNA). Biologická funkcia, ktorú vykonáva daná doména môže byť určená vlastnosťami aminokyselín v jej sekvencii, alebo v niektorých prípadoch môže byť dôležitejšia trojdimenzionálna štruktúra, do ktorej sa proteín po translácii poskladá, než samotné aminokyseliny tvoriace sekvenciu

Obr. 1.2: Primárna štruktúra sekvencie sú aminokyseliny, sekundárna štruktúra sú bloky aminokyselín s určitým nastavením väzieb a terciárna štruktúra hovorí aký útvar tvoria aminokyseliny v trojrozmernom priestore. Na obrázku je terciárna štruktúra pre doménu proteínu CDC13 zobrazená nástrojom *Jmol* (Jmol, 2012).

proteínu. Úseky proteínu, ktoré obsahujú dôležitú doménu, sú menej náchylné na zmenu v priebehu evolúcie, pretože často kódujú kľúčové vlastnosti proteínu. Táto vlastnosť sa dá využiť pri identifikovaní vzťahov medzi génmi v evolučne príbuzných organizmoch.

1.3 Evolúcia, ortológy

1.3.1 Evolučné procesy vplývajúce na genetickú informáciu

Počas evolúcie dochádza mutáciami k postupným zmenám v genetickom kóde, ktoré vedú k vývoju rôznych živočíšnych druhov. Mutácia životne dôležitého génu v organizme by mohla viesť k jeho znefunkčneniu a tým k úmrtiu bunky.

Duplikácia génu je často považovaná za hlavný mechanizmus, ktorý umožnuje diverzifikáciu druhov (Taylor and Raes, 2004). Pri delení bunky sa môže stať, že časť kódujúca niektorý gén sa skopíruje dvakrát a nová bunka bude obsahovať dve kópie tohto génu. Toto umožní kumuláciu mutácií v niektorej z kópií génu pri zachovaní funkčnosti niektorej z kópií. Kópia génu s mutáciami môže nadobudnúť novú funkcionalitu (kódovať iný

proteín), alebo sa funkcionalita pôvodného génu môže mutáciami rozdeliť medzi jeho dve kópie (Francino, 2005).

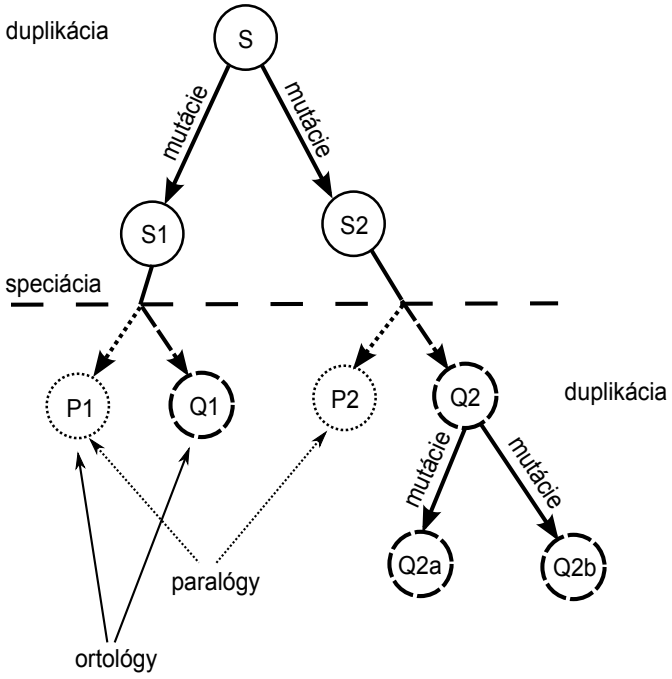
Pri delení bunky sa tiež môže stať, že časť DNA sekvencie kódujúca niektoré gény sa vynechá pri duplikácii genetickej informácie a dôjde k *strate* génov v novej bunke. Takýmito zmenami postupne dochádza k vývoju nových živočíšnych druhov z pôvodného spoločného predka. Gény, ktoré vznikli zo spoločného predka, sa nazývajú *homologické*.

1.3.2 Vzťahy homológie medzi génmi

Rozlišujeme dva typy homológie medzi génmi, ktoré sa líšia udalosťou, ktorá z pôvodného génu v spoločnom predkovi vytvorila dva rozdielne gény. *Ortológy* sú definované ako gény, ktoré vznikli následkom speciácie dvoch rozdielnych druhov z jedného predka (Fitch, 2000). Na tieto gény sa dá pozerať aj ako na „rovnaké gény“ v rôznych organizmoch a dá sa predpokladať, že vykonávajú rovnakú funkciu.

Dva gény sú *paralogické*, ak vznikli z jedného génu jeho duplikáciou (Fitch, 2000). Paralogické gény môžu tiež vykonávať rovnakú funkciu, ale keďže nie sú pod evolučným tlakom (ak sa jedna kópia mutáciami znefunkční, organizmus má stále jednu funkčnú verziu), môžu častejšie nadobudnúť novú funkciu a tým priniesť inováciu do génovej výbavy organizmu (Ohno et al., 1970; Francino, 2005). Paralogické gény môžu byť v dvoch rôznych genómoch (ak duplikácia nastala pred speciáciou), ale aj v tom istom genóme (ak duplikácia nastala po speciácii). Procesy, ktorým vznikajú ortologické a paralogické gény, sú znázornené na obrázku 1.3.

Zmapovať tieto vťahy medzi génmi, je užitočné, pretože tieto gény si počas evolúcie zachovali niektoré spoločné vlastnosti. To znamená, že informácie, ktoré sú platné o jednom géne z dobre preskúmaného organizmu, sa dajú použiť na predikciu rôznych vlastností jeho ortológu v inom organizme.



Obr. 1.3: Vzťahy paralógie a ortológie medzi génmi. Najprv sa v spoločnom predkovi duplikoval gén S na dve kópie (paralogické) *S1* a *S2*. Potom nastala speciácia a z jedného druhu sa vyvinuli dva (P a Q), každý so svojou verziou pôvodných génov *S1* a *S2*. Dvojice génov *P1*, *Q1* a *P2*, *Q2* sú ortologické. A nakoniec nastala duplikácia pre gén *Q2*, z ktorého vznikli gény *Q2a* a *Q2b*. Tieto sú navzájom paralogické a sú ortologické ku génu P2.

1.4 Rozdiely medzi kvasinkovými druhmi z pohľadu rozsahu dostupnosti dát

Biologické experimenty, ktorými sa získavajú informácie o genetických sekvenciách organizmov, boli objavené relatívne nedávno (Sanger and Coulson, 1975; Maxam and Gilbert, 1977). Kvôli ich finančnej aj časovej náročnosti bol výskum z počiatku zameraný na niektoré vybrané organizmy a až s rozvojom technológie začal počet osekvenovaných druhov rásť. Prvou kompletne osekvenovanou kvasinkou bola *Saccharomyces cerevisiae* (pekárske droždie), ktorá je (aj kvôli ekonomickým dôvodom) najintenzívnejšie študovaným kvasinkovým druhom. Následne boli ďalšie analýzy (anotácia génov v genóme, mapovanie proteínových interakcií a podobne) robené primárne na genóme tohto druhu. S rozvojom technológii boli osekvenované aj genómy ostatných kvasinkových druhov, avšak tieto genómy nie sú tak intenzívne skúmané ako *Saccharomyces cerevisiae*. Pre tieto druhy nie je k dispozícii tak veľký objem experimentálnych dát a mnohé ešte nemajú ani oanotovaný genóm.

Jednou z možností, ako sa vysporiadať s týmto nedostatkom, je využiť poznatky z dobre popísaných príbuzných druhov. Ak vieme odhadnúť funkciu konkrétneho génu v jednom organizme, dá sa predpokladať, že zodpovedajúca verzia tohto génu (jeho ortológ) v inom druhu si zachová rovnakú funkciu. Podobne sa dá uvažovať aj pri proteínových interakciách. Ak poznáme dva proteíny, ktoré interagujú v jednom organizme, je pravdepodobné, že budú interagovať aj ich ortologické verzie v iných druhoch. Takýmto spôsobom je možné analyzovať aj druhy, o ktorých nie je veľa dostupných dát. Problém pri tomto prístupe nastáva, ak sa pre niektorý známy dôležitý gén nenájde príslušný gén v skúmanom genóme. V takom prípade je potrebné zistiť či sa ho len nepodarilo nájsť, alebo či v danom genóme skutočne chýba. Formuláciou a riešeniu tohto problému z informatického hľadiska sa budeme venovať v tejto práci.

Kapitola 2

Metódy hľadania ortologických génov

Identifikovať navzájom ortologické gény v genómoch evolučne príbuzných organizmov je možné na základe rôznych indícií. Pretože vzťah ortológie medzi génmi vyplýva z ich evolučnej príbuznosti (vyvinuli sa mutáciami z jedného génu v spoločnom predkovi), ortologické gény majú často do veľkej miery podobnú genetickú sekvenciu. Takáto podobnosť však nie je nutným ani postačujúcim dôkazom ortologickej príbuznosti. Vysokú sekvenčnú podobnosť majú aj gény, ktoré vznikli duplikáciou (Fitch, 2000) a na druhú stranu ortologické gény mohli v priebehu evolúcie prejsť takými zmenami, že ich sekvencie nemajú štatisticky významnú zhodu.

Pre nájdenie vzťahu ortológie medzi génmi je preto nutné zobrať do úvahy ďalšie dáta ako napríklad pozície génov v ich genóme (Wapinski et al., 2007), či interakcie kódovaných proteínov s ostatnými proteínmi v danom organizme. Táto dodatočná informácia môže pomôcť napríklad odlíšiť paralógy od ortológov.

2.1 Sekvenčná podobnosť, zarovnania

Porovnanie dvoch genetických sekvencií je základným nástrojom analýz v bioinformatike. Ak sekvencie pochádzajú zo spoločného evolučného predka, dá sa predpokladať, že sa sekvencie líšia len v určitom počte mutácií, a teda že sú z väčšej časti rovnaké. Počas evolúcie sa v sekvencii predka mohli zmeniť jednotlivé bázy, časti sekvencie mohli byť vypustené alebo naopak mohli byť určité časti do sekvencie pridané. Vzhľadom na tieto

mutácie sa pri porovnávaní sekvencií uvažujú aj možné medzery.

2.1.1 Zarovnanie

Pri určovaní podobnosti dvoch sekvencií sa hľadá také ich *zarovnanie*, ktoré medzi nimi určí najväčšiu zhodu. Táto zhoda sa väčšinou vyjadruje cez skóre, ktoré danému zarovnaniu priradí zvolená *skórovacia schéma*. Skórovacia schéma každej zarovnanej dvojici báz určí skóre a skóre celej sekvencie sa získa súčtom skór pre všetky dvojice v zarovnaní. Skórovacie schémy môžu byť jednoduché (napríklad +1 za zhodu, -1 za nezhodu, -1 za zarovnanie bázy s medzerou), ale pretože za najlepšie by mali byť označené biologicky relevantné zarovnania, v praxi sa používajú komplexnejšie schémy (Dayhoff et al., 1978; Henikoff and Henikoff, 1992). Tieto schémy berú do úvahy dĺžky medzier v sekvencii alebo vlastnosti aminokyselín a s tým spojený dopad zmeny jednej aminokyseliny na inú. Napríklad ak sa na určitom mieste v jednej sekvencii nachádza leucín (L) a v druhej izoleucín (I), čo sú aminokyseliny s podobnými biologickými vlastnosťami, skóre takejto dvojice by malo byť kladné napriek tomu, že sú to dve rôzne aminokyseliny. Na presnejšie skórovanie rôznych dvojíc sa používa *skórovacia matica*.

Skórovacia matica

Prvky skórovacej matice popisujú skóre pre zarovnanie každej možnej dvojice písmen z abecedy (pre DNA 4 nukleotidové bázy, pre proteíny 20 aminokyselín) zarovnávaných sekvencií. Riadky aj stĺpce matice predstavujú jednotlivé písmená z používanej abecedy a tak napríklad prvok matice S_{AG} nad abecedou aminokyselín predstavuje skóre zarovnania dvojice alanín a glycín. Zmysluplne určiť hodnoty prvkov v tejto matici je kľúčové pre relevantné ohodnotenie zarovnania. V praxi sa používa viacero techník ako vypočítať takéto matice, napríklad PAM (Dayhoff et al., 1978) alebo BLOSUM (Henikoff and Henikoff, 1992).

Skórovacie matice BLOSUM (BLOcks of amino acid SUbstitution Matrix) vychádzajú z porovnania pravdepodobnosti výskytu aminokyselín v dvoch evolučne príbuzných sekvenciách a pravdepodobnosti výskytu aminokyselín v náhodnej sekvencii. Prvky v takejto

matici sa vypočítajú podľa vzorca:

$$S_{ij} = \log\left(\frac{p_{ij}}{q_i \cdot q_j}\right),$$

kde p_{ij} predstavuje pravdepodobnosť výskytu zarovnaného páru aminokyselín i a j, q_i a q_j predstavujú pravdepodobnosť náhodného výskytu aminokyseliny i respektíve j.

Uvažujeme zarovnanie dvoch sekvencií dĺžky $x_{1..n}$, $y_{1..m}$ bez medzier (zarovnanie bez medzier implikuje, že n a m sú rovnaké). Pravdepodobnosť náhodnej sekvencie sa vyjadrí ako súčin pravdepodobnosti výskytu aminokyselín na jednotlivých pozíciách v sekvencii. Takže pravdepodobnosť dvoch nezávislých náhodných sekvencií bude:

$$P(x, y|N) = \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{m} q_{y_i}$$

Pravdepodobnosť, že dve zarovnané aminokyseliny x_i a y_j vznikli nezávisle zo spoločného predka k označíme p_{ij}. Pravdepodobnosť celého zarovanania sekvencií x a y získame ako súčin pravdepodobností zarovnaných dvojíc:

$$P(x, y|Z) = \prod_{i=1}^{n} p_{x_i y_i}$$

Pomer pravdepodobnosti týchto dvoch modelov sa dá vyjadriť ako:

$$\frac{P(x, y|Z)}{P(x, y|N)} = \frac{\prod_{i=1}^{n} p_{x_i y_i}}{\prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}} = \frac{\prod_{i=1}^{n} p_{x_i y_i}}{\prod_{i=1}^{n} q_{x_i} q_{y_i}} = \prod_{i=1}^{n} \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

Zlogaritmovaním posledného výrazu násobenie prevedieme na sumu, ktorá sa ľahšie numericky počíta a jednotlivé sčítance predstavujú prvky zo skórovacej matice. Takto dostaneme logaritmické vyjadrenie pravdepodobnosti evolučnej príbuznosti sekvencií ako súčet skór pre jednotlivé dvojice aminokyselín zo zarovnania:

$$\log \frac{P(x, y|Z)}{P(x, y|N)} = \sum_{i=1}^{n} \log\left(\frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}\right)$$

Pravdepodobnosti p_{ij} dvoch zarovnaných aminokyselín získame z už známych zarovnaní normalízáciou počtu výskytov v už známych zarovnaniach.

Globálne a lokálne zarovnanie

Vo všeobecnosti sa uvažujú dva problémy zarovnania sekvencií. Pri *globálnom zarovnaní* je snaha nájsť podobnosť medzi sekvenciami ako celkami. Takéto porovnanie predpokladá približne rovnakú veľkosť sekvencií. Takýto prístup je vhodný napríklad na porovnávanie konkrétnych génov navzájom.

Ak očakávame podobnosť medzi sekvenciami len v určitom úseku, ako napríklad ak chceme identifikovať spoločné domény proteínu, alebo v prípade, že sekvencie majú výrazne rozdielnu dĺžku, hľadanie zarovnania sa sústredí na najlepšie zhodujúce sa úseky v oboch sekvenciách. Zarovnanie len určitých úsekov sa nazýva *lokálne zarovnanie*. Toto zarovnanie sa používa napríklad pri hľadaní jednotlivých génov v sekvencii celého genómu alebo hľadaní domén v proteíne.

Pri hľadaní týchto zarovnaní vychádzame z dobre študovaného informatického problému spoločných podpostupností reťazcov. Rovnako ako pre tento problém sa pri hľadaní oboch typov zarovnania používajú algoritmy založené na princípe dynamického programovania.

2.1.2 Needleman-Wunschov algoritmus

Needleman-Wunschov algoritmus (Needleman and Wunsch, 1970) nájde optimálne globálne zarovnanie pre dané dve sekvencie a dané skórovacie parametre (matica, penalizácia za medzery). Toto zarovnanie sa vyskladá pomocou použitia výsledkov z najlepších zarovnaní kratších podpostupností pôvodných dvoch sekvencií. Základná myšlienka spočíva v tom, že najlepšie globálne zarovnanie sekvencií dĺžky k získame z najlepšieho zarovnania dĺžky $k-1$ rozšírením o ďalší zarovnaný pár vybraný podľa najvyššieho skóre z nasledujúcich troch možností: pridať k zarovnaniu zarovnané nasledujúce prvky oboch sekvencií, pridať k zarovnaniu prvok prvej sekvencie zarovnaný s medzerou, pridať k zarovnaniu prvok druhej sekvencie zarovnaný s medzerou. Tieto tri možnosti sú znázornené na obrázku 2.1. Efektívnejšiu verziu tohto algoritmu popísal Gotoh (Gotoh, 1982).

Tento algoritmus má na vstupe sekvencie $X = x_{1..n}$, $Y = y_{1..m}$, ktorých prvky sú z

		F(i-1, j-1)	*+s(i,j)*			*F(i-1, j)*	*-d*			*F(i, j-1)*	*-d*
... T	A	x_{i-1}	x_i	... T	A	x_{i-1}	x_i	... A	x_{i-1}	x_i	-
... T	C	y_{j-1}	y_j	... C	y_{j-1}	y_j	-	... T	C	y_{j-1}	y_j

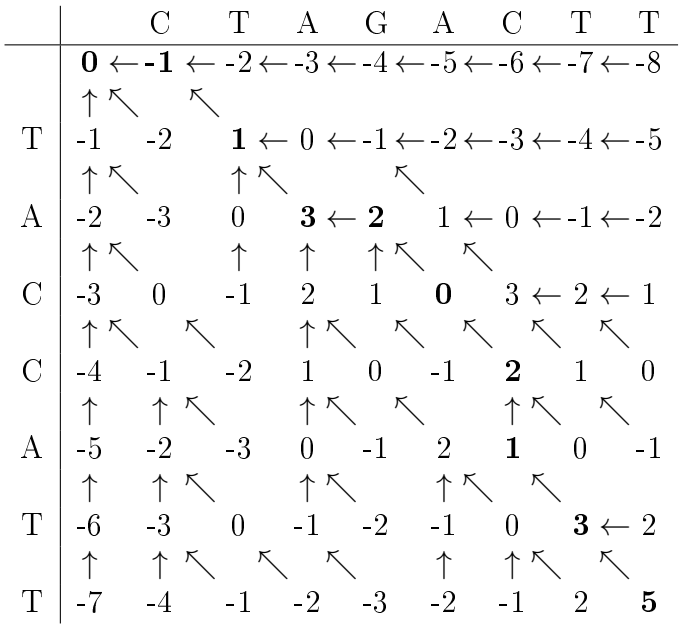
Obr. 2.1: Príklad troch možností pre zarovnanie podpostupností $x_{1..i}$, $y_{1..j}$. V prvom stĺpci je zarovnanie *F(i-1, j-1)* rozšírené zarovnaním dvojice aminokyselín x_i a y_j. V druhom a treťom stĺpci je aminokyselina x_i respektíve y_j zarovnaná s medzerou.

konečnej abecedy (aminokyseliny alebo nukleotidové bázy), ďalej skórovaciu funkciu *s(i,j)*, ktorá určí skóre pre dva zarovnané prvky i, j a parameter penalizácie za zarovnanie s medzerou d. Algoritmus vypočíta maticu F, ktorej prvky budú predstavovať skóre najlepšieho zarovnania podpostupností $x_{1..i}$, $y_{1..j}$. Skóre optimálneho zarovnania vstupných sekvencií X a Y bude obsahovať prvok matice *F(n,m)*. Skóre zarovnania dvoch prázdnych podpostupností sa bude rovnať *F(0,0) = 0*. Globálne zarovnanie postupnosti dĺžky k a prázdnej postupnosti je zarovnanie, ktoré obsahuje k medzier. Z toho vyplýva, že prvky matice *F(k,0)*a *F(0,k)* budú mať skóre *-kd*. Z takto inicializovanej matice sa vypočíta skóre pre ostatné prvky.

Zarovnanie podpostupností $x_{1..i}$, $y_{1..j}$ je možné dosiahnuť troma spôsobmi. Rozšírením zarovnania podpostupností dĺžky *i-1* a *j-1* o zarovnaný pár (x_i, y_j). Skóre takéhoto zarovnania je súčtom $s(x_i, y_j)$ a *F(i-1, j-1)*. Ďalšie dve možnosti pre zarovnanie je zarovnať prvok x_i (resp. y_j) s medzerou. V takom prípade sa použije zarovnanie so skóre *F(i-1,j)* (resp. *F(i,j-1)*) a od tohto skóre sa odčíta penalizácia za medzeru v zarovnaní. Z týchto troch možností sa vyberie to s najvyšším skóre a tak sa určí optimálne zarovnanie podpostupností $x_{1..i}$, $y_{1..j}$. V prípade, že dve možnosti majú rovnakú najvyššiu hodnotu, vyberie sa ľubovoľná z nich. Čiže prvok *F(i, j)* sa dá vypočítať podľa rekurentného vzťahu:

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases} \tag{2.1}$$

Na základe tejto rekurencie sa po riadkoch od ľavého horného rohu vypočítajú prvky matice F. V pravom dolnom rohu matice v prvku *F(n,m)* bude po ukončení algoritmu skóre optimálneho zarovnania vstupných postupností X a A. Pre rekonštrukciu tohto op-



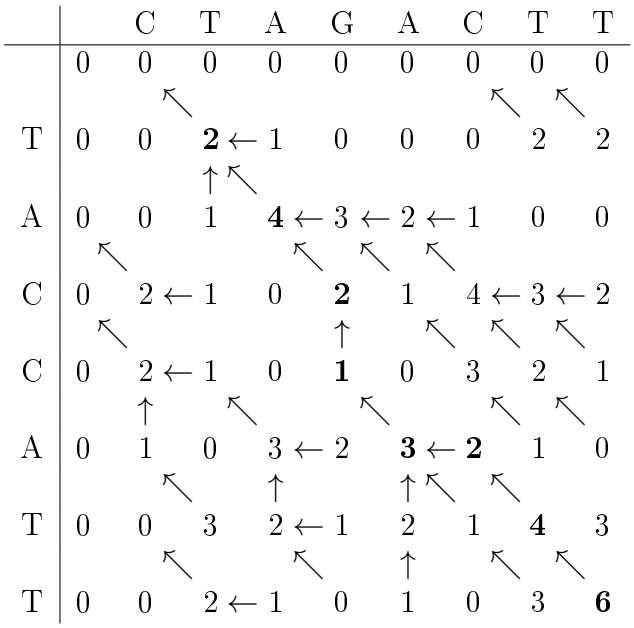
Obr. 2.2: Optimálne globálne zarovnanie je vyznačené hrubo. Skórovanie: +2 za zhodu, -2 za nezhodu, -1 za zarovnanie bázy s medzerou.

timálneho riešenia je nutné si počas výpočtu zapamätať pre každý prvok matice F, ktorá z troch možností sa použila na dosiahnutie najlepšieho skóre zarovnanie. Každé takéto rozhodnutie predstavuje jeden stĺpec zarovnania a určuje, či boli v danom stĺpci zarovnané prvky z postupností X a A, alebo bol zarovnaný prvok niektorej z postupností s medzerou. Podľa týchto zapamätaných výberov sa spätne od posledného stĺpca v zarovnaní vyskladá optimálne globálne zarovnanie X a A.

Príklad výpočtu globálneho zarovnania je na obrázku 2.2. Modifikáciou tohto algoritmu sa dajú hľadať lokálne zarovnania.

2.1.3 Smith-Watermanov algoritmus

Smith-Watermanov algoritmus (Smith and Waterman, 1981) vychádza z algoritmu pre globálne zarovnanie. Má rovnaké vstupné údaje (postupnosti X, A, skórovacia matica a penalizácia za medzery) a veľmi podobný mechanizmus. Základný rozdiel je v rekurencii na výpočet prvku matice F. Okrem troch pôvodných možností na výpočet hodnoty *F(i,j)*



Obr. 2.3: Optimálne lokálne zarovnanie je vyznačené hrubo. V tomto prípade je lokálne zarovnanie podmnožinou optimálneho globálneho zarovnania, nemusí to však byť pravidlom. Skórovanie: +2 za zhodu, -2 za nezhodu, -1 za zarovnanie bázy s medzerou.

pridáme možnosť dosadiť 0, ak by ostatné možnosti mali záporné skóre:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \tag{2.2}$$

Po tejto zmene všetky prvky matice, ktoré by mali zápornú hodnotu, budú nastavené na 0. To umožní začať či skončiť počítať skóre lokálneho zarovnania na ľubovoľnom mieste v sekvenciách. Kvôli tejto zmene sa na 0 nastavia tiež všetky hraničné prípady *F(k,0)* a *F(0,k)*. Takto v matici bude len niekoľko nenulových prvkov, ktoré budú predstavovať rôzne lokálne zarovnania podpostupností. Prvok matice s najvyššou hodnotou bude obsahovať skóre optimálneho lokálneho zarovnania. Pre získanie tohto najlepšieho zarovnania budeme od tohto prvku spätne podľa zapamätaných výberov v rekurzívnom výpočte skladať zarovnanie. Rekonštrukcia sa ukončí, keď narazíme na prvok matice s hodnotou 0.

Táto verzia algoritmu dokáže nájsť zarovnanie s najvyšším skóre. Príklad výpočtu zarovnania týmto algoritmom je na obrázku 2.3. Pri dlhých sekvenciách však môžu aj ostatné lokálne zarovnania s dostatočne dobrým skóre predstavovať relevantný výsledok

(napríklad ak majú dva proteíny v sekvenciách viac spoločných domén). Tieto ďalšie zarovnania sa dajú ľahko získať po odfiltrovaní prekrývajúcich sa zarovnaní. Hľadanie viacerých lokálnych zarovnaní je popísané v práci (Waterman and Eggert, 1987).

Uvedené algoritmy pre globálne a lokálne zarovnanie nájdu optimálne riešenie s časovou zložitosťou *O(nm)* (vyplnenie matice $n \times m$). V prípade rozsiahlych biologických sekvencií (genóm *S.cerevisiae* má cez 12 miliónov bázových párov), sú tieto algoritmy v praxi ťažko použiteľné. Preto je nutné uchýliť sa k heuristickým metódam, ktoré nezabezpečujú nájdenie optimálneho riešenia, ale sú vhodnejšie pre praktické účely.

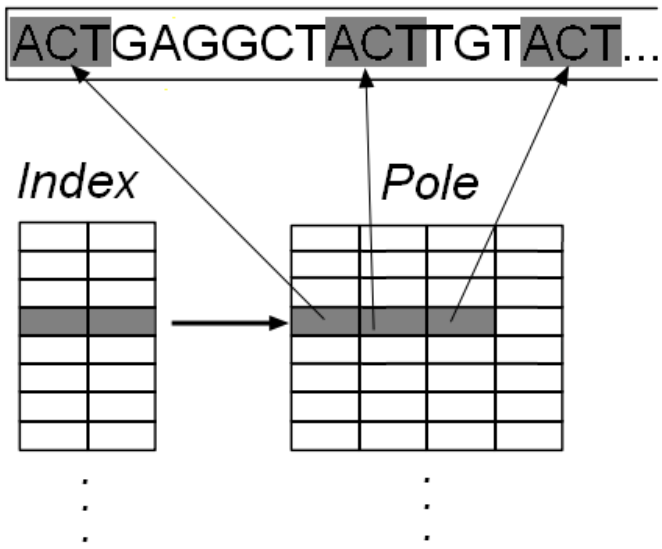
2.1.4 BLAST - Basic Local Alignment Search Tool

Sada programov BLAST (Altschul et al., 1990) využíva na hľadanie signifikantných lokálnych zarovnaní podľa danej skórovacej schémy heuristický prístup a snaží sa dosiahnuť optimálne výsledky, ktoré by dosiahli algoritmy dynamického programovania. Programy BLAST sady umožnujú hľadať zarovnania medzi jednou zdrojovou sekvenciou (*dotaz*) a sekvenciami z danej *databázy*. Dotaz aj sekvencie v databáze môžu byť z nukleotidovej (4 prvky) alebo aminokyslinovej (20 prvkov) abecedy. Pre danú kombináciu abecied sekvencií pre dotaz a databázu sa použije špecifický program zo sady (BLASTN, BLASTP, TBLASTN a podobne), ktorý implementuje správny prístup k danej kombinácii (napríklad rozdielne skórovacie matice pre aminokyseliny a nukleotidové bázy). Programy BLASTu hľadajú lokálne zarovnania so skóre, ktoré je vyššie ako určitá hranica, pre všetky kombinácie (*dotazu*) a sekvencie z databázy.

Algoritmus BLASTového hľadania vychádza z predpokladu, že zarovnania s najväčším skóre obsahujú krátke úseky úplnej zhody alebo úseky s veľmi veľkým skóre podobnosti, medzi sekvenciami. Hľadanie sa sústredí na tieto kratšie úseky, *jadrá*, ktorých rozšírením potom nájdeme čo najlepšie zarovnanie. Samotný algoritmus pre hľadanie lokálnych zarovnaní postupuje v troch krokoch:

- `Určenie jadier podsekvencií v dotaze`.

  V prvom kroku sa analyzuje vstupná sekvencia dotazu s parametrom dĺžky jadra



Obr. 2.4: Príklad vyhľadávacej tabuľky pre jadrá dĺžky 3.

w a prahom pre skóre T. Pre každú podpostupnosť dotazu dĺžky w sa vytvorí zoznam sekvencií, ktoré majú pri zarovnaní s touto podpostupnosťou skóre väčšie ako daný prah T. Tieto sekvencie sa budú následne hľadať v databázových sekvenciách. V praxi sa pre nukleotidové sekvencie používa dĺžka slova $w = 11$ a pre aminokyselinové sekvencie 3. Tento rozdiel vyplýva najmä z rozdielnych skórovacích matíc pre tieto rôzne abecedy. Pre efektivitu ďalšieho hľadania sa pre tieto sekvencie ukladajú ich pozície vo *vyhľadávacej* tabuľke (obrázok 2.4). Táto má pre každé jadro index ukazujúci do komprimovaného (kvôli úspore pamäti) poľa ukazovateľov na pozíciu jadra v dotaze. Takáto dátová štruktúra umožňuje rýchlu lokalizáciu pozície nájdeného jadra v sekvencii dotazu.

- `Hľadanie jadier v sekvenciách v databáze`.

  Po určení jadier zo sekvencie dotazu sa hľadajú s nimi zhodné podsekvencie v sekvenciách databázy pomocou algoritmu Wilbur and Lipman (1983). Nájdené zhodujúce sa kandidátske podsekvencie následne poslúžia ako základ pre rozšírenie na lokálne

zarovnania s vysokým skóre.

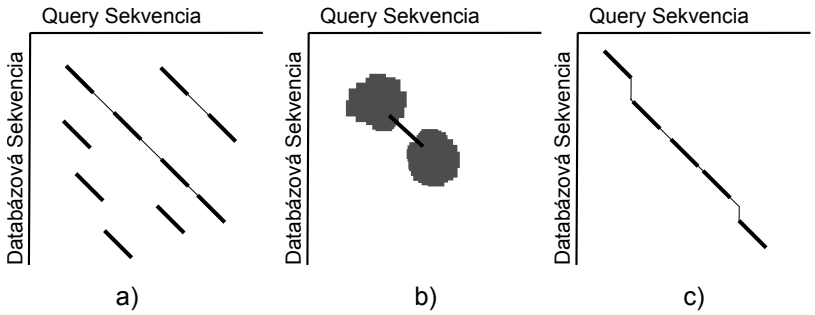
- `Rozšírenie nájdených jadier na čo najlepšie zarovnania`.

  Pre každú nájdenú kandidátsku podsekvenciu sa vypočíta rozdiel medzi jeho začiatočnou pozíciou v dotaze a jeho pozíciou v cieľovej sekvencii. Tento rozdiel určuje, ktoré kandidátske podsekvencie môžu byť spolu v jednom zarovnaní bez medzier (nachádzali by sa na rovnakej diagonále v matici dynamického programovania). Ak nie sú tieto jednotlivé podsekvencie od seba vzdialené viac ako parameter A, BLAST sa pokúsi ich spojiť do jedného väčšieho zarovnania bez medzier. Ak bude mať toto zarovnanie skóre väčšie než ďalší paratemer *S1*, považuje sa rozšírenie za úspešné.

  Následne sa skúsia dosiahnuť lepšie lokálne zarovnania s pomocou medzier. Vypočítané zarovnanie bez medzier bude základom pre hľadanie lepšieho zarovnania. Vybrané zarovnania budú rozširované na obidvoch koncoch, ale iba pokiaľ skóre takého zarovnania bude väčšie ako skóre zarovnania bez medzier mínus *orezávací* parameter D. Takýmto spôsobom sa efektívne BLAST vyhne počítaniu príliš veľkého počtu možností, pretože keď rozšírený úsek sekvencií má nízke skóre, jeho ďalším rozšírením skóre rýchlo klesne pod nulu. Jednotlivé kroky tohto postupu sú znázornené na obrázku 2.5.

  Z takto rozšírených *hitov* sa vyberú tie so skóre väčším ako ďalší parameter *S2*. Tieto zarovnania sa zoradia podľa najlepšieho skóre a zrekonštruujú sa kvôli zobrazeniu.

BLAST je heuristický algoritmus, neprehľadáva všetky možnosti, takže zarovnanie, ktoré zobrazí nemusí byť optimálne. Senzitivitu algoritmu, ako aj jeho výpočtovú náročnosť ovplyvňuje viacero parametrov. Asi najdôležitejším je dĺžka hľadaného jadra w. V prípade, že by bolo slovo príliš krátke, sa zväčšuje pravdepodobnosť náhodného hitu medzi dotazom a databázou. To znamená veľa kandidátskych podsekvencií na rozširovanie, čím sa výrazne zvyšuje čas behu algoritmu. Naopak, ak by w bolo príliš veľké, mohlo by sa stať, že by algoritmus neodhalil zhodné podpostupnosti dĺžky w - *1*, ktoré by mohli po rozšírení tvoriť zarovnanie s veľmi vysokým skóre. Takže algoritmus by nenašiel niektoré biologicky



Obr. 2.5: Priklady rozšírenia kandidátskych podsekvencií. a) spájanie podsekvencií podľa diagonál. b) hľadanie s orezávaním, šedo vyznačené oblasti okolo vybranej podsekvencie, ktoré sa prehľadávajú. c) rozšírenie s medzerami.

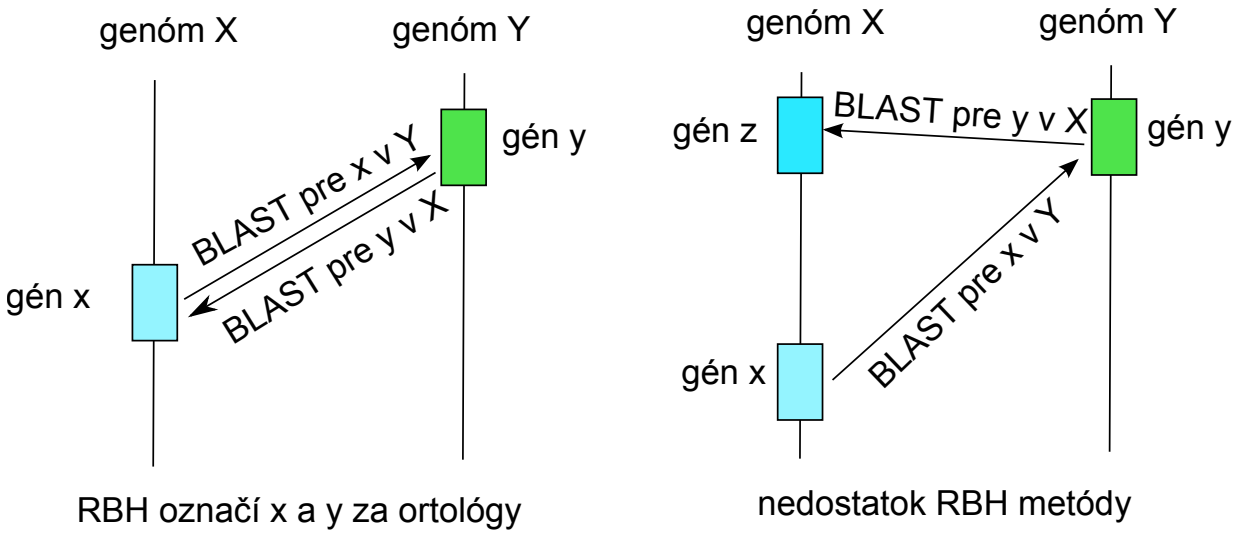
relevantné zarovnania. Rovnako zle nastavený orezávací parameter D môže spôsobiť, že BLAST prestane prehľadávať predčasne a minie tak niektoré zarovnania. Takisto ostatné používané parametre (*T, A, S1, S2*) ovplyvňujú senzitivitu algoritmu a jeho časovú zložitosť. Ďalšie detaily o vlastnostiach a implementácii tohto algoritmu môžno nájsť napríklad v prácach Cameron et al. (2004) alebo Chang (2005).

Pre posúdenie relevantnosti výsledku nájdeného hľadaním nám iba skóre nestačí, pretože pri malej dĺžke sekvencie by sa mohlo stať, že zarovnanie dvoch sekvencií s daným skóre by sa podarilo nájsť aj medzi sekvenciami, ktoré nie sú pribuzné. Preto sa k výsledkom okrem skóre počíta aj *E-value*, čo je očakávaný počet zarovnaní s daným alebo lepším skóre, ktoré by sa našli medzi dvoma čisto náhodnými sekvenciami.

2.2 Hľadanie ortologických sekvencií pomocou BLASTu

2.2.1 Reciprocal best hit (RBH) metóda

Najpriamočiarejšie použitie BLASTu na hľadanie ortológov je metóda *reciprocal best hit* - najlepších vzájomných hitov (výsledkov hľadania). Podstatou tejto metódy je predpoklad, že dva ortologické gény v dvoch organizmoch, budú navzájom najpodobnejšie, pretože vznikli zo spoločného predka. Gén x z genómu X je najlepším vzájomným hitom (*rbh*) pre gén y z genómu Y, ak najlepší výsledok hľadania BLASTom s query x v genóme Y je gén y a zároveň pri obrátenom hľadaní s query génu y v genóme X bude najlepší BLAST hit



Obr. 2.6: Vľavo je zobrazené úspešné hľadanie ortológou pomocou metódy RBH. V pravej časti obrázku je prípad keď táto metóda neidentifikuje dvojicu ortologických génov x a y

gén x. Identifikácia ortológou pomocou RBH metódy je znázornená na obrázku 2.6. Táto metóda je široko používaná, viď. napríklad Rivera et al. (1998); Hirsh and Fraser (2001).

Obmedzenia RBH metódy a jej zlepšenia

Samotná RBH metóda nezabezpečuje nájdenie správnej dvojice ortológov. V prípade, že sú ortologické gény príliš evolučne vzdialené (majú príliš rozdielne sekvencie), môže sa stať, že nebude nájdený žiadny signifikantný hit alebo najlepšie hodnotený hit nebude ortológ ku hľadanému génu (viď. obrázok 2.6). Hľadanie BLASTom nie vždy vráti evolučne najbližší gén v hľadanom organizme (Koski and Golding, 2001) . Pri hľadaní ortológov metódou RBH problém najčastejšie predstavujú paralógy hľadaného génu. BLAST môže v jednom smere hľadania nájsť správny ortológ, ale pri hľadaní v opačnom smere vyjde ako najlepší hit paralóg génu a metóda RBH zavrhne takúto dvojicu. Tento problém sa dá riešiť dodatočnou analýzou nájdených riešení podľa rôznych kritérií.

2.2.2 Reciprocal smallest distance (RSD) metóda

Jedným z navrhnutých spôsobov zlepšenia RBH metódy je navrhnutý v práci Wall, Fraser, Hirsh Wall et al. (2003). Myšlienka tohto prístupu spočíva v použití BLASTu na získanie

malej množiny vysoko podobných sekvencií, ktoré sa porovnajú na základe vzájomnej evolučnej vzdialenosti. Dve evolučne najbližšie sekvencie z tohto porovnania sa vyhlásia za ortológy.

Pri metóde RSD sa najprv BLASTom pre hľadanú sekvenciu x z genómu X získa množina hitov *H1* s vysokým skóre z genómu Y. Prvky tejto množiny sú spätne zarovnané k x (algoritmom CLUSTALW - Larkin et al. (2007)) a pre zarovnané dvojice, ktorých zarovnané časti presahujú určitý prah z dĺžky celého zarovnania, sa vypočíta programom PAML (Yang, 2000) najpravdepodobnejší počet substitúcií aminokyselín, ktoré oddeľujú x a danú hit sekvenciu. Z takto prepočítaných sekvencií z množiny *H1* sa zoberie, len sekvencia y s najmenšou evolučnou vzdialenosťou, ktorá sa následne spätne použije na hľadanie BLASTOM v genóme X. Získaná množina hitov *H2* sa spracuje rovnako ako *H1* pri opačnom smere hľadania. Ak z tohto zpracovania vyjde ako sekvencia z najmenšou evolučnou vzdialenosťou k a pôvodne hľadaná sekvencia x, dvojica x a y sa označí za ortologické sekvencie.

Ortologické páry, ktoré sú odhalené metódou RSD, ale nie pomocou RBH, predstavujú pravdepodobne prípady, keď RBH v niektorom smere vyhlásila za najlepší hit paralóg k hľadanému génu. Výsledky ďalšej analýzy (Moreno-Hagelsieb and Latimer, 2008) však poukazujú na väčšiu mieru chýb v prístupe RSD, takže tento prístup nemusí vždy poskytnúť výhodu oproti RBH.

Tieto metódy sú jadrom viacerých databáz skupín ortologických proteínov. Jednotlivé databázy sa líšia rôznym spracovaním výsledkov z RBH metód, ktorým sa snažia adresovať niektoré problémy týchto metód.

2.2.3 Databáza COG

Databáza COG (Clusters of Orthologous Groups of proteins) (Tatusov et al., 2000) predstavuje snahu o fylogenetickú klasifikáciu proteínov z 21 kompletných genómov baktérií aj eukariotických organizmov. Skupiny týchto proteínov sú tvorené na základe predpokladaných ortologických vzťahov medzi nimi. Pri konštrukcii týchto skupín sa vychádza zo sekvenčnej podobnosti proteínov určenej na základe BLAST hľadania. Základnou myšlien-

kou je vytvorenie skupiny pre aspoň 3 proteíny z rozdielnych genómov, ktoré sa navzájom podobajú viac než na ktorýkoľvek iný proteín z daných genómov. Konštrukcia týchto skupín postupuje nasledovne:

1. Vykoná sa BLAST hľadanie s každým proteínom z každého genómu ako dotazom.

2. Identifikujú a zlúčia sa predpokladané paralógy - proteíny z toho istého genómu, ktoré sú podobné navzájom viac, než voči ľubovoľnému proteínu z iného genómu.

3. Detekujú sa trojuholníky proteínov z rôznych genómov, ktoré sú navzájom najlepšie hity (pri zohľadnení už identifikovaných paralógov). To znamená, že proteín x z genómu X má navzájom najlepší hit s a z genómu A, A má v genóme Z recipročne najlepší hit proteín z a z je vzájomne najlepší hit pre x z genómu X.

4. Trojuholníky proteínov, ktoré majú spoločnú hranu, sa spoja do väčších skupín - zhlukov COG.

5. Vykoná sa analýza každého COGu na elimináciu falošne pozitívnych priradení a určenie skupín obsahujúcich viacdoménové proteíny. Tieto proteíny mohli byť nesprávne zaradené do jedného zhluku kvôli veľkej podobnosti niektorých ich domén aj keď nie sú v ortologickom vzťahu. Sekvencie týchto proteínov sa rozdelia podľa domén a kroky 1-4 sa zopakujú na tieto podsekvencie, čím sa dosiahne zaradenie jednotlivých domén do COG zhlukov.

6. Následne sa analyzujú veľmi veľké COGy, ktoré obsahujú proteíny z väčšiny alebo zo všetkých genómov. Tieto zhluky sa po preskúmaní zarovnaní medzi jednotlivými ich proteínmi, môžu rozdeliť na dve alebo viac finálnych COG zhluky.

Z tohto postupu vyplýva, že minimálny počet proteínov z rôznych genómov v jednom zhluku je 3. Takto zoskupené proteíny predstavujú výpočtovú predpoveď založenú na sekvenčnej podobnosti a je potrebné podrobiť tieto výsledky ďalšej analýze na overenie biologickej relevantnosti. Ďalším z problémov COG metódy je nerozlišovanie medzi evolučne starými a novšími duplikáciami.

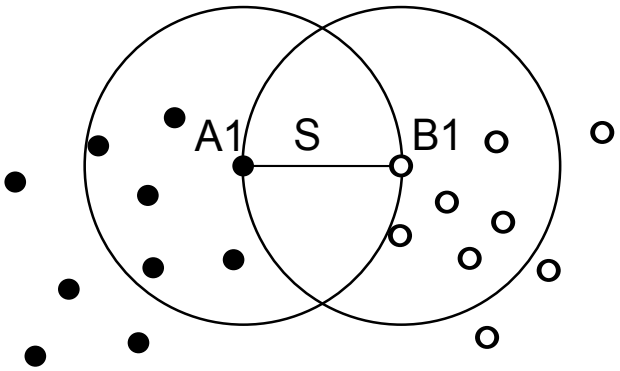
2.2.4 INPARANOID

Na podobnom princípe ako COG pracuje aj databáza INPARANOID (Remm et al., 2001). Algoritmus na určovanie ortológov INPARANOID však berie do úvahy časový sled evolučných udalostí duplikácie a speciácie. Ak nastane duplikácia génu po jeho speciácii, obe kópie génu sú zároveň ortologické ku génu z druhého organizmu, ktorý vznikol speciáciou. Takéto paralógy sa označujú ako *in-paralógy*. Naopak paralogické gény, ktoré vznikli pred speciáciou sa nazývajú *out-paralógy*. INPARANOID sa pri určovaní vzťahu ortológie snaží určiť aj in-paralógy k najlepšiemu vzájomnému hitu a odlíšiť ich od out-paralógov. Algoritmus na vstupe dostane dva kompletné zoznamy proteínových sekvencií z dvoch organizmov a ako výstup vypočíta zoznam skupín ortologických proteínov v týchto organizmoch. Kedže je snaha identifikovať in-paralógy a zahrnúť ich do správnych ortologických skupín, tieto skupiny môžu pozostávať z viacerých proteínov z oboch organizmov. Zároveň sa vypočíta miera dôveryhodnosti pre tieto pridané ortológy. Konštrukcia týchto skupín postupuje nasledovne:

1. Určia sa sekvenčné podobnosti medzi všetkými sekvenciami z organizmov A a B a tiež medzi sekvenciami jednotlivých organizmov navzájom (A-B, A-A, B-B, B-A). Na toto určenie sa použije BLAST hľadanie (*all-in-all*).

2. Nájdu sa vzájomné najlepšie hity medzi proteínmi organizmov A a B. Tieto dvojice budú tvoriť hlavný ortologický pár v zkonštruovaných ortologických skupinách.

3. Ku všetkým takto nájdeným dvojiciam sa pridajú ich predpokladané in-paralógy, ktoré sa určia pre každý organizmus zvlášť. Tento krok algoritmu je znázornený na obrázku 2.7. Pri identifikovaní in-paralógov sa vychádza z predpokladu, že in-paralógy sú proteíny organizmu, ktoré sú podobné na proteín z hlavného ortologického páru viac než na ľubovoľný iný proteín v druhom organizme. Na sekvenčnú podobnosť medzi hlavným ortologickým párom proteínov A1 a B1 sa dá pozerať ako na vzdialenosť medzi sekvenciami týchto proteínov. Vzdialenosť medzi hlavným ortologickým párom označíme S. In-paralógy k proteínu A1 sú tie proteíny z organizmu A, ktorých



Obr. 2.7: Proteíny organizmu A sú vyznačené čiernymi bodmi, proteíny organizmu B bielymi. Vzdialenosť S medzi hlavným ortologickým párom A1-B1 určuje hranice pre zodpovedajúce skupiny in-paralógov.

vzdialenosť od proteínu A1 je menšia ako vzdialenosť S (vzdialenosť medzi A1 a B1). In-paralógy k proteínu B1 sa nájdu analogicky medzi proteínmi organizmu B. Takto sa vytvoria dve skupiny in-paralógov, jedna pre každý organizmus.

4. K in-paralógom sa vypočíta miera dôveryhodnosti. Táto miera vyjadruje, ako veľmi je in-paralóg podobný na hlavný proteín skupiny, kvôli ktorému bol pridaný medzi ortológy.

5. Prekrývajúce sa skupiny proteínov sa roztriedia, prípadne spoja do jednej väčšej skupiny na základe skór podobností proteínov v týchto skupinách.

6. K samotným skupinám proteínov sa priradia miery dôveryhodnosti. Ak je rozdiel medzi hlavným proteínom v skupine a druhým väčší očakáva sa, že výber hlavného proteínu skupiny bol správny a miera dôveryhodnosti je väčšia ako keď by boli alternatívne proteíny v skupine s veľkou podobnosťou k hlavným ortológom.

Na zlepšenie tohto hľadania môže byť na vstupe pridaný zoznam proteínov z tretieho refenčného organizmu, ktorý môže pomôcť odfiltrovať ortológy so nízkym skóre podobnosti medzi A a B.

Na podobnom princípe ako INPARANOID je postavený aj algoritmus OrthoMCL (Li et al., 2003), ktorý navyše umožňuje zoskupovanie proteínov pre viaceré organizmy.

2.3 Vzťah ortológie určený podľa pozície v genóme

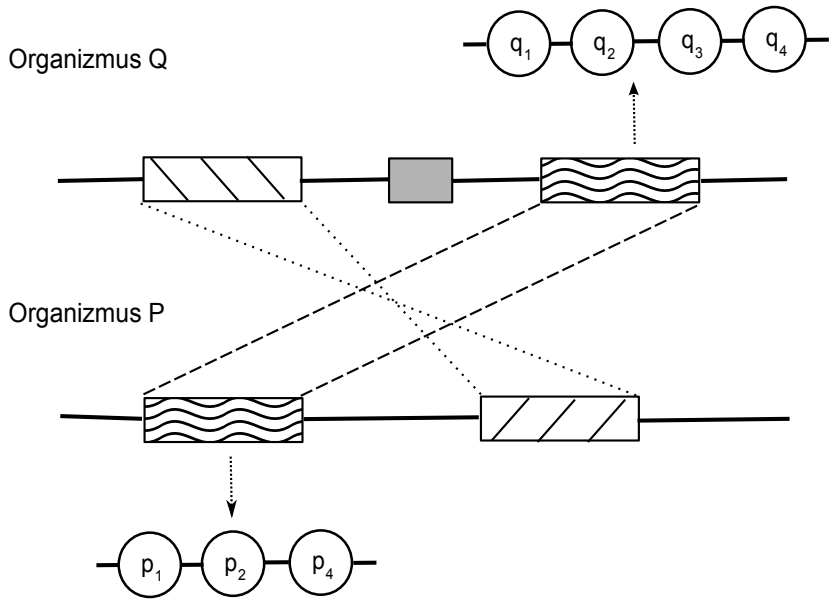
2.3.1 Syntenické regióny genómov

Pozícia génu v genóme môže poslúžiť ako dodatočná informácia, ktorá pomôže správne identifikovať ortologické gény z kandidátov získaných na základe sekvenčnej podobnosti. Úseky genómov dvoch evolučne príbuzných organizmov, ktoré obsahujú príbuzné gény, sa označujú za *syntenické*. Príklad takýchto úsekov je na obrázku 2.8. Počas evolúcie sa môže stať, že sa úsek sekvencie presunie na iné miesto v genóme. Takéto úseky môžu obsahovať aj viac génov. Ak takáto zmena v genóme nastane až po speciácii, úseky obsahujúce ortologické proteíny môžu byť na rozdielnych miestach v chromozómoch jednotlivých organizmov, ale kontext genómu môže pomôcť určiť, či dva sekvenčne podobné proteíny vznikli speciáciou. Pri analýze BLAST hitov pri RBH metóde je potrebné odlíšiť hľadané ortológy od paralógov, ktoré vznikli duplikáciou ešte pred speciáciou. V genómoch okolo predpokladanej dvojice ortologických proteínov sa očakáva prítomnosť ďalších podobných ortologických dvojíc. Naopak pre neortologické proteíny je malá pravdepodobnosť, že budú mať navzájom podobný kontext vo svojich genómoch. Overenie tohto predpokladu poskytne ďalšie kritérium pre správnosť predpokladaného vzťahu ortológie. Použitie tohto kritéria je súčasťou *Best Unambiguous Set (BUS)* metódy.

2.3.2 Best Unambiguous Set (BUS) metóda

Táto metóda je popísaná v (Kellis et al., 2004). Umožnuje nájsť korešpondujúce gény medzi dvoma organizmami na základe sekvenčnej podobnosti a informácii o genómovom kontexte génov. Tato metóda nadväzuje na metódu RBH a zhlukovanie metódov COG. Jej princíp spočíva v hľadaní množín génov, pre ktoré platí, že všetky najlepšie zhodujúce sa gény s niektorým génom množiny sú tiež v tej istej množine. Zároveň množina neobsahuje žiadny najlepší hit pre gén mimo množiny. Na začiatku sa vytvoria množiny génov podľa najlepších hitov nad nejakou hranicou a následne sa aplikovaním rôznych kritérií z pôvodných množín vyextrahujú množiny spĺňajúce vyššie uvedenú podmienku.

Na reprezentáciu problému nájdenia týchto množín sa použije grafová štruktúra. Vr-



Obr. 2.8: Príklad syntenických regiónov. Dva organizmy P a Q sa vyvinuli zo spoločného predka a majú na genóme úseky, ktoré navzájom korešpondujú. Takéto úseky obsahujú viac navzájom ortologických génov (p1 - q1, p2 - q2, p4 - q4), ale kvôli zmenám počas evolúcie (napríklad reorganizácia genómu) nemusia byť tieto úseky v rovnakej časti genómu.

choly budú predstavovať gény každého organizmu a ováhované hrany medzi nimi predstavujú sekvenčnú podobnosť získanú z výsledkov BLASTu. Na základe týchto informácií sa vytvorí bipartitný graf (gény jednoho organizmu budú spojené hranou len s génmi druhého organizmu). Z tohto grafu sa budú eliminovať hrany, pre získanie ortologických skupín. Najprv sa však neorientované hrany nahradia orientovanými(každá orientovaná hrana má rovnakú váhu ako neorientovaná pred ňou.) V prvom kroku sa odstránia tie hrany, ktorých skóre je menšie ako 80% z váhy najlepšie hodnotenej hrany vychádzajúcej z toho istého vrchola. Množina hrán, ktorá ostane po tomto kroku, sa značí M_{80}. Tento krok odfiltruje jasne evolučne rozdielne sekvencie a výrazne zníži súvislosť grafu. Komponenty tohto grafu budú predstavovať množiny ortologických proteínov. Po prvom kroku rozlíšenia na základe sekvenčnej podobnosti sa tieto množiny analyzujú podľa genomického kontextu.

Podgrafy vytvorené v predchádzajúcom kroku sa použijú na vytvorenie *blokov zachovanej synténie* založenými na fyzickej vzdialenosti medzi génmi v genóme. Napríklad, ak v grafe sú hrany (x_1, y_1) a (x_2, y_2), v genóme X sú v blízko seba umiestnené gény x_1 a x_2 a

v genóme A to iste platí pre gény y_1 a y_2, vytvorí sa syntenický blok $B = (x_1, y_1), (x_2, y_2)$. Následne sa tieto bloky použijú na orezanie hrán, ktoré spájajú gény mimo toho istého bloku. Takže ak by gén y_3 nebol fyzicky blízko génom y_1 a y_2 hrana z medzi ním a génom x_1 alebo x_2 by sa odstránila. V praxi sa na orezávanie hrán používajú bloky obsahujúce aspoň 3 gény. Pre konštrukciu blokov je dôležitý parameter d, ktorý určuje maximálnu vzdialenosť medzi génmi v rámci ktorej sa považujú za fyzicky blízke. V uvedenej práci bol tento parameter nastavený na 20 kilobáz, čo predstavuje zhruba 10 génov.

Po tomto orezaní sa zo zvyšných podgrafov vyberú tie, ktoré spĺňajú podmienku pre best unambiguous set. To znamená, že pre každý vrchol z množiny je jeho najlepší hit tiež v množine a žiadny vrchol mimo množiny nemá svoj najlepší hit vnútri tejto množiny. Také množiny sa skonštrujú orezaním na M_{100} (len najlepšie vychádzajúce hrany z vrcholov) a nájdením súvislých komponentov v tomto grafe. Tieto množiny predstavujú predpokladané skupiny ortologických génov.

2.4 Hľadanie ortologických sekvencií pomocou profilov a motívov

Predchádzajúce metódy sa zameriavali na určenie ortológie na základe podobnosti medzi dvojicami sekvencií. Takýmto prístupom boli identifikované vzťahy ortológie medzi génmi rôznych druhov, čo umožňuje skúmať spoločné vlastnosti rôznych ortologických verzií jedného génu. Pre každý organizmus môže verzia génu obsahovať odlišnosti špecifické pre daný druh. Pri hľadaní ortológov podľa sekvencie konkrétneho druhu sa pri zarovnávaní berú do úvahy aj tieto špecifické časti, čo môže znížiť skóre relevantného hitu. Z tohto dôvodu pre hľadanie ortológov v evolučne vzdialenejších organizmoch, ktorých sekvencie divergovali výraznejšie, nemusí stačiť ani postupné porovnanie kandidátskej sekvencie so všetkými známymi verziami génu na správnu identifikáciu ortológu. V takýchto prípadoch je potrebné zamerať porovnanie len na tie časti génu, ktoré sú zachované medzi všetkými známymi verziami, pretože sa dá očakávať, že práve tieto časti sa budú vyskytovať v géne aj vzdialenejšieho organizmu.

2.4.1 Profily sekvencií

Na určenie úsekov zachovaných medzi viacerými sekvenciami sa používa *viacnásobné zarovnanie sekvencií*. Toto zarovnanie je analogické k zarovnaniu dvojice sekvencií. V tomto prípade sa tiež hľadá zarovnanie s najlepším skóre podľa daných parametrov (skórovacia matica, penalizácia za medzery), ale pre viac ako 2 sekvencie. Nájsť optimálne zarovnanie viacerých sekvencií je NP-úplný problém (Wang and Jiang, 1994) a v praxi sa na hľadanie týchto zarovnaní používajú heuristické metódy. Medzi najrozšírenejšie implementácie týchto metód patria nástroje CLUSTALW (Larkin et al., 2007) a MUSCLE (Edgar, 2004).

Z výstupov týchto nástrojov sa dajú identifikovať zachované úseky medzi jednotlivými sekvenciami. Každý stĺpec zarovnania predstavuje prvky (prípadne medzery) vyskytujúce sa na danom mieste v jednotlivých sekvenciách. Zhoda prvkov v stĺpci pre väčšinu sekvencií naznačuje, že aj pri ďalších sekvenciách patriacich do rovnakej skupiny sa dá očakávať na danej pozície práve rovnaký prvok, ako u väčšiny zarovnaných sekvencií. Viac stĺpcov so signifikantnou zhodou za sebou môže predstavovať evolučne zachovaný úsek sekvencie, ktorý je charakteristický pre danú skupinu sekvencií. Naopak stĺpce zarovnania, v ktorých nie je zrejmá zhoda medzi sekvenciami rôznych druhov, pravdepodobne neobsahujú informáciu podstatnú pre klasifikovanie sekvencie. Viacnásobné zarovnanie teda poskytuje informáciu o tom, ktoré pozície v sekvencii sú rozhodujúce a aké prvky sa na nich očakávajú pre sekvenciu patriacu do danej skupiny. Na reprezentovanie spoločných čŕt skupiny génov sa následne používajú štruktúry, ktoré umožňujú rozdielne skórovanie zhody medzi danou sekvenciou a modelom v závislosti na pozícii. Na dosiahnutie takejto reprezentácie sa používajú napríklad pozične špecifické skórovacie matice (*position specific scoring matrix - PSSM*) (Gribskov et al., 1987) alebo *profilové skryté Markovovské modely* (*profile Hidden Markov Model - HMM*) (Krogh et al., 1994). PSSM, ako z názvu vyplýva, je modifikáciou skórovacej matice z časti 2.1.1, avšak na rozdiel od skórovacej matice prvok *S(a, p)* v takejto matici označuje skóre výskytu prvku a z danej abecedy na pozícii p v danej sekvencii. Takouto maticou môžeme reprezentovať jednotlivé zachované časti sekven-

cie, takéto vyjadrenie však neberie do úvahy prípadné medzery v sekvencii. Tento problém riešia *profilové HMM*.

2.4.2 Profilové HMM

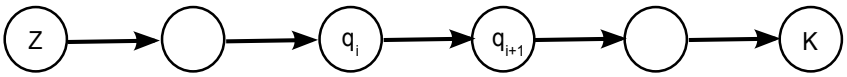
Vo všeobecnosti skryté Markovovské modely sú pravdepodobnostné modely, na ktoré sa dá pozerať ako na generujúce automaty. Tieto automaty majú konečnú množinu stavov, v ktorých sa generuje sekvencia. Prechody medzi stavmi majú danú pravdepodobnosť rovnako, ako vygenerovanie konkrétneho prvku sekvencie v danom stave.

Prívlastok skrytý vyplýva z toho, že pozorovaná je len vygenerovaná sekvencia. Postupnosť stavov, v ktorých sa automat pri generovaní nachádzal, ostáva skrytá. Rôzne sekvencie môžu byť modelom vygenerované rôznymi spôsobmi, ktoré majú rôznu pravdepodobnosť. Parametre modelu, ako sú pravdepodobnosti prechodov a emisií prvkov sekvencie, nesú informáciu o vlastnostiach sekvencií, ktoré tento model dokáže generovať s veľkou pravdepodobnosťou.

Jedno z možných použití HMM je, že nastavením týchto parametrov sa dajú vytvoriť modely, ktoré reprezentujú určitú skupinu sekvencií. Následne sa takýto model dá použiť na vypočítanie pravdepodobnosti príslušnosti danej sekvencie k modelovanej skupine tak, že sa vypočíta pravdepodobnosť vygenerovania tejto sekvencie týmto modelom. Profilové HMM sú špeciálny prípad skrytých Markovovských modelov, ktoré svojou architektúrou umožňujú z viacnásobného zarovnania vyjadriť spoločné vlastnosti evolučne príbuzných biologických sekvencií.

Profilové skryté Markovovské modely vytvoria z viacnásobného zarovnania príbuzných sekvencií systém na ohodnocovanie sekvencií podľa výskytu jednotlivých prvkov na každom mieste v sekvencii. Na základe tohto ohodnotenia sa dá posúdiť, či ohodnotená sekvencia patrí do skupiny príbuzných sekvencií, ktoré boli použité na tvorbu modelu. Na výpočet pravdepodobností prechodov medzi stavmi a emisie prvkov sa použijú jednotlivé stĺpce zo vstupného zarovnania.

Na profilové HMM sa dá pozerať ako na rozšírenie PSSM o možnosť uvažovať medzery. Samotné PSSM sa dá vyjadriť ako jednoduchý HMM, ktorý pozostáva zo stavov zoradených

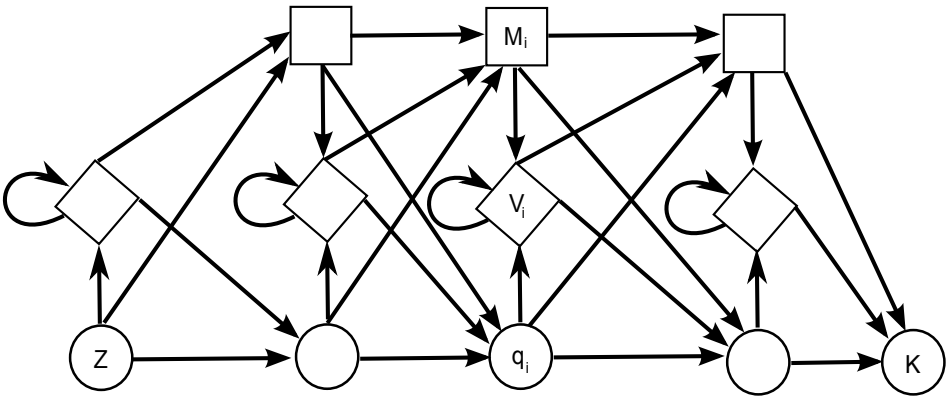


Obr. 2.9: Skrytý Markovovský model, ktorý neuvažuje medzery. Model má začiatočný stav Z , konečný stav K a zarovnávacie stavy q_i. v každom stave je len jedna možnosť prechodu do ďalšieho stavu.

do reťaze. Z každého stavu q_i sa dá pohnúť len do stavu q_{i+1} po vygenerovaní znaku a. Pravdepodobnosť vygenerovania každého znaku abecedy v danom stave q_i zodpovedá pravdepodobnosti výskytu tohto znaku na pozícii i. Tieto stavy reprezentujú zarovnané prvky sekvencií a budeme ich označovať ako *zarovnávacie* stavy. Prechody medzi týmito stavmi sú dané vždy len jednou možnosťou, takže pravdepodobnosť prechodu medzi stavmi bude vždy 1. Takýto model neuvažuje medzery v sekvencii a funguje ako PSSM pri hodnotení kandidátskej sekvencie. Schéma tohto modelu je na obrázku 2.9.

Rozšírenie modelu o medzery musí poskytovať možnosť reprezentácie vloženia či straty úseku sekvencie. Vloženie úseku predstavuje časť sekvencie, ktorá nezodpovedá ničomu v modelovaných sekvenciách. Možnosť generovania takýchto častí v sekvencii je realizovaná prostredníctvom *vkladacích* stavov, do ktorých sa dá dostať po každom zarovnávacom stave. Z takého stavu sa pokračuje do nasledujúceho zarovnávacieho stavu, prípadne do stavu modelujúceho stratu úseku sekvencie. Pre modelovanie tejto straty je potrebné pridať do modelu možnosť preskočiť určitý počet zarovnávacích stavov. Z tohto dôvodu sú do modelu pridané *mazacie* stavy. Každý mazací stav predstavuje vynechanie jedného prvku. Do takéhoto stavu sa dá dostať z príslušného (podľa pozície) zarovnávacieho, vkladacieho alebo mazacieho stavu a pokračovať do nasledujúcich stavov ľubovoľného typu. Schéma rozšíreného HMM je na obrázku 2.10. Takáto štruktúra je typická pre profilové HMM a ako prví ju predstavili (Haussler et al., 1993) a (Krogh et al., 1994).

Pre získanie modelu, ktorý vyjadruje z daného viacnásobného zarovnania spoločné črty zarovnaných sekvencií, je nutné určiť parametre modelu, ako sú pravdepodobnosti prechodov a emisií pre jednotlivé stavy a tiež dĺžka modelu. Pod dĺžkou modelu sa rozumie počet zarovnávacích stavov a problém spočíva v rozhodnutí, ktoré stĺpce zarovnania obsahujú dostatočnú zhodu, aby boli reprezentované takýmto stavom. Toto môže byť



Obr. 2.10: Profilový skrytý Markovovský model. Model má začiatočný stav Z , konečný stav K, zarovnávacie stavy q_i, mazacie stavy M_i, vkladacie stavy V_i.

rozhodnuté rôznymi heuristickými pravidlami; napríklad stĺpec zarovnania, ktorý obsahuje medzeru pre viac ako polovicu sekvencií, bude modelovaný vkladacím stavom a nie zarovnávacím stavom. Pravdepodobnosti pre emisie sa dajú získať zo vstupného viacnásobného zarovnania spočítaním výskytov prvkov sekvencie v jednotlivých stavoch. Pravdepodobnosti prechodov medzi stavmi sa tiež určujú z tohto zarovnania spočítaním prechodov medzi jednotlivými stavmi modelu.

Takýto prístup dáva dobré výsledky pre veľké množstvo zarovnaných sekvencií, ale v prípade, že je trénovacích sekvencií málo, môže byť pre prechody a emisie, ktoré sa v nich nevyskytujú, táto pravdepodobnosť nastavená na nulu. Takto nastavená pravdepodobnosť nie je žiadúca, pretože takto by sa dané prechody zakázali pre všetky sekvencie posudzované takýmto modelom. Najjednoduchšie sa tento problém dá riešiť započítaním pseudo-výskytu pre všetky prechody a emisie, ale existujú rôzne ďalšie prístupy, ako lepšie získavať pravdepodobnosti pre profilové HMM, ktoré sú popísané v práci Durbin et al. (1998).

Po natrénovaní modelu je možné použiť ho na detekciu ďalších podobných sekvencií. Sekvencia sa dá klasifikovať na základe pravdepodobnosti, že bola vygenerovaná daným modelom. Táto pravdepodobnosť sa dá získať štandardným *forward* algoritmom pre HMM (Durbin et al., 1998). Pre posúdenie relevantnosti danej sekvencie je nutné porovnať výslednú pravdepodobnosť s pravdepodobnosťou vygenerovania danej sekvencie náhodným

modelom.

2.4.3 HMMER a Pfam

Profilové HMM sú implementované v rôznych verejne prístupných nástrojoch, ktoré umožňujú ich praktické použitie. Takýmto široko používaným nástrojom je napríklad HMMER (Eddy, 1998), ktorý poskytuje celý rad možností ako použiť profilové HMM (hľadať hity pre HMM v databáze sekvencií, emitovať z modelu sekvencie a podobne). Tento nástroj bol s úspechom použitý v práci Gabaldón et al. (2005), v ktorej sa pomocou neho podarilo detegovať vzťahy ortológie medzi komponentami enzýmu NADH a proteínmi bez určenej funkcie, v rôznych organizmoch. Použítím tohto nástroja bola tiež vytvorená databáza Pfam (Finn et al., 2007), ktorá obsahuje modely rodín príbuzných proteínov, vytvorených z viacnásobného zarovnania, a ďalšie údaje (popis domén, štruktúry a podobne) pre proteíny z týchto rodín.

2.4.4 Motívy a MEME

Modely vo vyššie popísanej databáze Pfam popisujú podobnosť medzi sekvenciami v rodine na ich celej dĺžke. Pri riešení niektorých úloh je však potrebné identifikovať jednotlivé kratšie zachované úseky medzi príbuznými sekvenciami a reprezentovať každý takýto úsek zvlášť. To umožní hľadať v iných sekvenciách len tieto časti, alebo klasifikovať podľa nich už známe sekvencie. Z biologického hľadiska tieto úseky môžu predstavovať štrukturálne (aminokyseliny z daného úseku sa poskladajú do rovnakej trojrozmernej štruktúry) alebo funkčné celky, ktoré sa vyskytujú v proteínoch príbuzných organizmov, ako napríklad rovnaké väzobné domény v príbuzných proteínoch. Takýto zachovaný úsek medzi sekvenciami sa nazýva *motív*. Tieto motívy sa väčšinou reprezentujú pomocou vyššie spomínaných pozične špecifických skórovacích matíc, ktoré určujú, na ktorej pozícii a s akou váhou sa v danom úseku očakáva daná aminokyselina. Motív môžeme graficky reprezentovať logom, ktoré pre každú pozíciu obsahuje aminokyseliny zobrazené s veľkosťou úmernou pravdepodobnosti výskytu na danej pozícii. Príklad takéhoto loga zobrazujúceho sekvenčný motív je na obrázku 2.11.

Obr. 2.11: Grafické zobrazenie sekvenčného motívu vytvoreného pomocou programu MEME (Bailey et al., 2006). Na osi x je pozícia v sekvenčnom motíve, na osi y je zlogaritmovaná pravdepodobnosť výskytu zobrazenej aminokyseliny na danej pozícii. Aminokyseliny sú podľa ich chemických vlastností rozdelené do štyroch skupín. Farbou aminokyseliny je na obrázku znázornená jej príslušnosť do jednej z týchto skupín.

Sada takýchto motívov sa dá použiť na hľadanie sekvencií obsahujúcich rovnaké domény. Prítomnosť týchto motívov v určitej sekvencii môže indikovať jej príbuznosť ku skupine sekvencií, z ktorých boli tieto motívy vytvorené.

Na identifikáciu motívov v sekvenciách a na hľadanie s nimi je takisto viac verejne prístupných nástrojov. Jedným z takýchto nástrojov je napríklad balík programov The MEME Suite (Bailey et al., 2006). Program MEME (Multi EM for Motif Elicitation) (Bailey and Elkan, 1994) identifikuje v daných nezarovnaných sekvenciách zadaný počet motívov s najnižšou E-value (stredná hodnota počtu výskytov daného motívu v náhodnej sekvencii). Štandardné nastavenie pre počet hľadaných motívov je tri. Tento program hľadá len motívy bez medzier a reprezentuje ich pomocou PSSM. Nájdené motívy je možné následne použiť ako vstup do programu MAST (Motif Alignment and Search Tool), ktorý umožňuje hľadanie týchto motívov v databáze sekvencií (Bailey and Gribskov, 1998). Prácu s motívmi, ktoré obsahujú medzery, umožňujú ďalšie programy (GLAM, GLAM2SCAN), ktoré sa nachádzajú v aktuálnejších verziách balíka programov The MEME Suite (Frith et al., 2008).

2.5 Fylogenetická informácia

Pre správne určenie vzťahov ortológie je nutné odlíšiť v sekvenčne podobných génoch paralógy od ortológov. Algoritmus SYNERGY (Wapinski et al., 2007) rieši tento problém pomocou dodatočnej informácie o evolučnej histórii organizmov a génov kódujúcich proteíny v týchto organizmoch. *Fylogenetických strom* je grafovou reprezentáciou predpokladanej evolučnej histórie daných organizmov (alebo génov) a určuje sa analýzou sekvencií v súčasnosti existujúcich organizmov. Evolučná história génov a samotných organizmov je rôzna, pretože gény mohli vznikať v priebehu evolúcie nielen pri speciácii organzimov, ale aj procesom duplikácie. Z toho vyplýva, že aj gény z rovnakého organizmu môžu mať s časti rozdielne fylogenetické stromy. Fylogenetický strom zachytáva evolučné procesy, ktoré viedli k vyvinutiu jednotlivých druhov (génov) zo spoločného predka. Vrchol v strome predstavuje organizmus (gén), pričom v listoch sú súčasné organizmy (gény súčasných organizmov) a vnútorné vrcholy tohto stromu predstavujú predpokladaných vyhynutých evolučných predkov (ich gény). Hrany v strome určujú, ktoré organizmy (gény) sa vyvinuli z jedného spoločného predka.

Algoritmus SYNERGY má na vstupe okrem zoznamu génov kódujúcich proteíny pre jednotlivé organizmy aj fylogenetický strom pre skúmané organizmy. Vybudovaním fylogenetických stromov pre jednotlivé gény sa vyrieši problém odlíšenia paralógov a ortológov. Tieto stromy určujú, ktoré gény vznikli prostredníctvom ktorých procesov (duplikácia či speciácia), a tak sú vzťahy medzi génmi (paralógia, ortológia) odvodené po vybudovaní týchto stromov z ich štruktúry. Samotné určovanie vzťahu homológie medzi génmi je v tomto algoritme riešené na základe už uvedených metód porovnania sekvenčnej podobnosti a tiež pomocou informácie o umiestnení génov v genóme (syntenické úseky). Pri rekoštrukcii fylogenetického stromu génov v smere od listov ku koreňu sa podľa vstupnej informácie z fylogenetického stromu organizmov, reorganizujú skupiny predpokladaných ortologických proteínov. Po zostrojení kompletného stromu génov sú definované ortologické skupiny, ktoré rozlišujú paralógy od ortológov na základe vypočítaného stromu. Takýmto spôsobom sú efektívne katalogizované proteíny viacerých organizmov podľa vzťahu ortológie.

Kapitola 3

Hľadanie domén kombináciou sekvenčných a štrukturálnych motívov

Domény v ortológoch majú podobné sekvencie, keďže vznikli evolučnými procesmi zo spoločného predka. V metódach popísaných v predchádzajúcej kapitole boli určované vzťahy ortológie na základe podobnosti v primárnej štruktúre. Avšak pre zachovanie biologickej funkcie môže byť dôležitejšia trojdimenzionálna štruktúra, do ktorej sa proteín po translácii poskladá, než samotné aminokyseliny tvoriace sekvenciu proteínu. V takých prípadoch sekvenčná podobnosť nie je dostatočne zachovaná a na identifikovanie ortologických domén je potrebné zobrať do úvahy ďalšie informácie o štruktúre proteínu. V tejto kapitole predstavíme prístup, ktorý používame v našej práci na hľadanie takýchto domén.

3.1 Hľadanie štrukturálnych motívov

V proteíne, ktorý je poskladaný v priestore, interagujú navzájom aj aminokyseliny, ktoré sú od seba značne vzdialené v rámci sekvencie. Takéto väzby môžu spájať jednotlivé úseky v sekundárnej štruktúre do väčších útvarov (napríklad viac β-listov spojených do β-barelu), takže tieto závislosti predstavujú dôležitú informáciu o štruktúre proteínu. Takéto útvary v štruktúre proteínu sa nazývajú *štrukturálne motívy* a ich identifikovanie umožňuje zvýšiť citlivosť vyhľadávania domén so silnejšie zachovanou štruktúrou.

Pri hľadaní štrukturálnych motívov v proteínoch stojíme pred problémom, ako skombinovať dostupné informácie o štruktúre proteínov s metódami z kapitoly 2 využívajúcimi

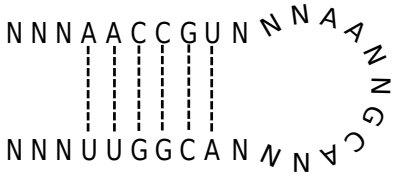
sekvencie aminokyselín na hľadanie podobnosti medzi proteínmi. Je nutné zadefinovať spôsob, akým budú všetky tieto informácie reprezentované a vyvinúť efektívne metódy, ktoré budú využívať túto reprezentáciu na identifikovanie motívov v proteínovych sekvenciách. Vo všeobecnosti sa tento problém dá riešiť dvoma rôznymi spôsobmi.

3.1.1 Automatické metódy hľadajúce štrukturálne motívy

Prvou možnosťou je rozšíriť modely používané na zachytenie spoločných čŕt v primárnej štruktúre pre príbuzné domény. K pôvodným reprezentáciám (napríklad profilové HMM) sa pridajú časti, ktoré vyjadrujú závislosti medzi vzdialenými aminokyselinami v sekvencii. Tieto závislosti nie sú zachytené v skrytých Markovovských modeloch, pretože v nich sú prechodové hrany ohodnotené len na základe dvoch susednych stavov.

Program SMURF (Menke et al., 2010) generalizuje Markovovské modely na Markovovské náhodné polia, ktoré umožňujú vyjadriť pravdepodobnosť prechodu do nového stavu v závislosti od viacerých predchádzajúcich stavov. V štandardných profilových HMM sa pravdepodobnosť vygenerovania aminokyseliny a_j v stave q_j vypočíta na základe predchádzajúceho stavu q_{j-1}. V programe SMURF sa výpočet pravdepodobnosti pre zarovnávacie stavy zodpovedajúce spárovaným úsekom β-listov rozšíri o interagujúcu aminokyselinu a_k. Pravdepodobnosť interakcie medzi vzdialenými aminokyselinami v β-listoch je teda možné vyjadriť výrazom β-strand$[a_j, a_k, q_j, q_k]$, kde a_j a a_k sú aminokyseliny vygenerované v stavoch modelu q_j a q_k. Tento výraz sa počíta len pre zarovnávacie stavy q_j a q_k, ktoré zodpovedali interagujúcim aminokyselinám v proteínoch použítých na natrénovnie modelu. Pridanie tohto výrazu zvýši výpočtovú náročnosť použitia tohto modelu, ale zároveň umožní rozoznávať štrukturálne motívy pozostávajúce najmä z β-listov.

Ďalším spôsobom ako reprezentovať β-listové štrukturálne motívy je pomocou metód, ktoré využívajú rôzne druhy gramatík, ako napríklad Ranked Node Rewriting gramatika (Mamitsuka and Abe, 1994), Range Concatenation gramatika (Chiang et al., 2006) alebo Multitape Context-Free gramatika (Waldispühl et al., 2006). Tieto gramatiky špecifikujú interakcie medzi aminokyselinami v stromoch odvodenia a pre vstupnú sekvenciu je potrebné nájsť najpravdepodobnejší strom odvodenia.



Obr. 3.1: Štrukturálny motív hairpin sa skladá z jedného stemu a jedného loopu. Písmenom N sa označuje možnosť výskytu ľubovoľnej nukleotidovej bázy na danom mieste.

Uvedené metódy štruktúru, ktorú chcú reprezentovať, určujú z trénovacích príkladov automaticky, výpočtom štatistických vlastností vzťahov medzi aminokyselinami vo vstupných sekvenciách. Takýto postup nemusí priniesť uspokojivé výsledky v prípade, že nie je k dispozícii dostatočne veľa dobrých trénovacích príkladov a natrénovaný model nezachytí želané štrukturálne vlastnosti hľadanej domény. Preto sme sa v našej práci rozhodli pre iný prístup, ktorý sa zameriava na reprezentáciu známych vlastností hľadanej štruktúry.

3.1.2 Metódy využívajúce špecifikované vlastnosti štrukturálnych motívov.

Druhou možnosťou ako hľadať štrukturálne motívy určitej domény je využiť informácie o jej vlastnostiach dostupné z rôznych biologických experimentov. Z týchto informácií môžu odborníci ručne zostaviť reprezentáciu domény, ktorá sa zameria na dôležité vlastnosti štrukturálneho motívu.

Náš prístup je motivovaný analogickým problémom hľadania štrukturálnych motívov v RNA sekvenciách. Pri hľadaní RNA génov je sekundárna štruktúra RNA kľúčová, pretože má veľký vplyv na biologickú funkciu sekvencie. V prípade RNA génov sa osvedčil prístup pomocou ručne vytvorených *deskriptorov* (Jimenez et al., 2012), ktoré popisujú dôležité črty hľadaného RNA génu. RNA je jednovláknová molekula, ktorej bázy sa viažu sami zo sebou, pričom vytvárajú zväčša komplementárne interagujúce páry (A - U , C - G). Vďaka týmto bázovým párom je možné priamočiaro popísať vzory v sekundárnej štruktúre RNA, ako motívy pozostávajúce z úsekov spárovaných a nespárovaných báz (*stemy* a *loopy*). Príkladom je jednoduchý štrukturálny motív *hairpin*, ktorý je na obrázku 3.1.

Deskriptory pre RNA charakterizujú hľadaný štrukturálny motív ako postupnosť úsekov

```
L1 S1 L2 S1 L3

L1 3:4
S1 6:6 AACCGU:ACGGUU
L2 8:14 AANNCGCA
L3 3:4
```

Obr. 3.2: Deskriptor pre štrukturálny motív hairpin z obrázku 3.1.

spomínaných dvoch typov, pričom sú pre jednotlivé úseky špecifikované ďalšie očakávané parametre (minimálna, maximálna dĺžka, obmedzenia na primárnu sekvenciu a podobne). Obmedzenia na primárnu sekvenciu sú reprezentované pomocou regulárnych výrazov, ktoré určujú aké nukleotidové bázy musí úsek obsahovať. V prípade spárovaných úsekov toto obmedzenie určuje aj navzájom interagujúce nukleotidové bázy. Na obrázku 3.2 je uvedený príklad takéhoto deskriptora pre štrukturálny motív hairpin z obrázku 3.1. Štrukturálne motívy popísané takýmito RNA deskriptormi sa následne dajú vyhľadávať v RNA sekvenciách. RNA sekvenciu obsahuje takýto motív ak v nej vieme určiť pozíciu pre začiatok motívu takú, že nukleotidové bázy od tejto pozície vyhovujú obmedzeniam špecifikovaným v deskriptore pre hľadaný motív.

Pre tvorbu analogických deskriptorov pre proteínové domény je potrebné vysporiadať sa s viacerými problémami, ktoré vyplývajú z rozdielnych vlastností RNA molekúl a proteínov. Regulárne výrazy sú príliš reštriktívne na zachytenie sekvenčnej informácie spoločnej pre danú doménu v ortologických proteínoch a preto je vhodnejšie vyjadriť podmienky na sekvenciu pomocou profilov. RNA molekuly sa skladajú zo 4 rôznych nukleotidových báz, ktoré sa navzájom pri interakciách dopĺňajú v preferovaných dvojiciach, zatiaľ čo proteíny pozostávajú z až 20 druhov aminokyselín, ktoré majú často podobné chemické vlastnosti (napr. leucín a izoleucín). Preto je problematické určiť interakcie v rámci poskladanej sekvencie a podmienky na párovanie aminokyselín sa nedajú zapísať jednoduchými obmedzeniami na primárnu sekvenciu ako tomu je v prípade RNA deskriptorov. Je teda nutné definovať spôsob ako ohodnotiť potenciálne interakcie medzi vzdialenými aminokyselinami. Kvôli týmto rozdielom je potrebné nájsť nielen vhodnejší spôsob

reprezentácie týchto vlastností pre proteínové domény, ale aj algoritmus dokáže vyhodnotiť do akej miery vyhovuje sekvencia aminokyselín takto reprezentovaným vlastnostiam.

Deskriptor nebude vhodný pre každú doménu, zvýšenú citlivosť hľadania môže priniesť tento prístup pre domény, v ktorých je štruktúra výraznou charakteristikou a sekvencia nie je dostatočne konzervovaná pre použitie štandardných metód. V ďalšej časti 3.3 popíšeme prípadovú štúdiu pre proteínoch asociovaných s telomérickými procesmi, ktorá poukazuje na niektoré nevýhody použitia štandardných metód pri hľadaní ortológov pre tieto proteíny. Následne v časti 3.4 na základe týchto výsledkov navrhneme deskriptor pre proteíny z tejto skupiny, ktorý môže pomocť zvýšiť citlivosť vyhľadávania ich ortológov.

3.2 Prípadová štúdia: proteíny telomérického komplexu

3.2.1 Telomérický komplex

Teloméry sú nukleoproteínové komplexy umiestnené na oboch koncoch lineárnych chromozómov, ktoré vykonávajú v bunkách veľmi dôležitú funkciu: chránia konce lineárnych chromozómov pre degradáciou, zabraňujú tomu, aby boli rozpoznávané ako dvojvláknové zlomy, inhibujú komponenty zapojené do reparačných mechanizmov a tým zabezpečujú stabilitu a celistvosť chromozómov (Lodish, 2003; Nosek and Tomáška, 2008).

Pri procese delenia bunky dochádza počas replikácie ku skracovaniu repetitívnych úsekov na koncoch chromozómov. Tento mechanizmus v bunkách predstavuje akési molekulárne hodiny, pretože po skrátení telomérických úsekov pod určitú minimálnu dĺžku nie je možná ďalšia bezchybná replikácia a dochádza k úmrtiu bunky. Zároveň v niektorých bunkách (napríklad kmeňové, či rakovinové) je aktívny enzým *telomeráza*, ktorý predlžuje konce chromozómov a tým umožňuje bunke ďalšie delenie. Telomeráza sa skladá z niekoľkých proteínov a z krátkej RNA sekvencie. Táto RNA sekvencia predstavuje vzor, podľa ktorého sa predlžujú repetitívne úseky. Fungovanie telomerázy ovplyvňujú rôznym spôsobom ďalšie proteíny, ktoré môžu podporovať alebo inhibovať jej aktivitu. Príťažlivosť tohto telomérického komplexu pre výskum spočíva v jeho dôležitej úlohe pri procese starnutia a vo vývoji niektorých ochorení, napr. rakoviny (Blackburn et al., 2006).

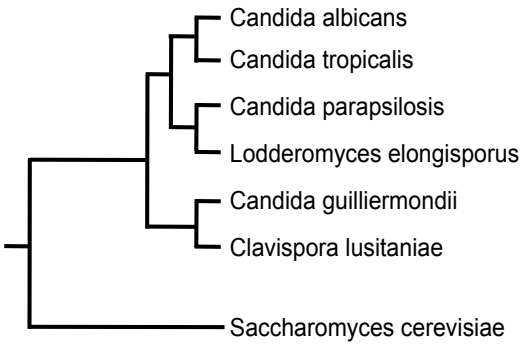
Proteíny asociované s telomérami, telomerázou a procesom údržby dĺžky telomér.
DEP1, FUN12, NUP60, SWD1, RFA1, HEK2, POL12, TEL1, MEC1, SWD3, PBP2, SHG1, RIF1, DCC1, MRC1, HTL1, HCM1, SIR2, RAD59, RPN6, YDL118W, RPN5, CDC9, CDC13, RAD57, KCS1, CDC34, STN1, RRP8, HPR1, HST4, SIR4, HDA2, SUM1, XRS2, MUS81, RPT3, SDC1, YDR532C, MCM3, YEL057C, RAD51, RPN12, RAD6, RAD54, SUA5, KEM1, MCM6, RTF1, YGR042W, UPF3, TEL2, SRB5, BUD32, OPI1, SRB2, NMD2, SET1, THP2, EST3, MET18, MLP2, PRI1, MPS3, RFA3, GON7, POL32, RAD27, MLP1, TEN1, BRE2, RSC58, STM1, IFH1, EST1, TOP3, MEC3, EST2, RSC2, CDC73, RIF2, RAD52, PIF1, ZDS2, NAM7, NPL6, YKU80, STO1, SPT21, SGS1, MRE11, MTF1, SAP30, ZDS1, YKU70, PRE5, IMP4, POL1, YAF9, RAP1, RAD50, RFA2, HST1, NDJ1, HRT1, YOR008C-A, EXO1, RAT1, RPT5, ELG1, NFI1, NPT1, RFM1, YPL041C, TBF1, BEM4, DAP1, CSM4, LEA1, PRE2, RPN7

Tabuľka 3.1: Zoznam 121 proteínov z *Saccharomyces cerevisiae* asociovaných s telomérickým komplexom použitých na hľadanie v genómoch iných kvasiniek. (Tomáška, 2009)

S aktivitou telomérického komplexu bolo v kvasinke *Saccharomyces cerevisiae* asociovaných vyše 200 rôznych proteínov (Shachar et al., 2008). Pri porovnaní s inými proteínovými sekvenciami medzi druhmi je možné pozorovať rýchlu evolúciu týchto proteínov napriek tomu, že vykonávajú v bunkách rovnakú esenciálnu úlohu a očakávateľná by bola vyššia zachovanosť sekvencií (Linger and Price, 2009). Rýchla evolúcia týchto proteínov znižuje šance identifikácie ortológov týchto proteínov v iných organizmoch len na základe sekvenčnej podobnosti. Na týchto skupine týchto proteínov sme testovali citlivosť metód pracujúcich so sekvenčnou podobnosťou. Výsledky uvádzame v nasledujúcej časti.

3.2.2 Hľadanie ortológov pomocou sekvenčnej zhody

Pre hľadanie ortológov pre telomérické proteíny kvasinky *Saccharomyces cerevisiae*v genómoch rôznych kvasiniek (uvedené v tabuľke 3.2) sme použili metódy popísané v kapitole 2. Metóda Reciprocal Best Hit (RBH) označí za ortológy dva proteíny z dvoch rôznych organizmov ak sa pri hľadaní podobnosti navzájom označia za najlepšie nálezy. Metódu RBH sme použili pomocou programu BLAST, profilové HMM pomocou nástroja HMMER (Eddy, 1998) a sekvenčné motívy pomocou nástrojov MEME (Bailey and Gribskov, 1998). Fylogenetický strom evolučnej príbuznosti pre skúmané druhy na obrázku 3.3 zob-



Obr. 3.3: Fylogenetický strom pre analyzované druhy kvasiniek.

Skúmané Organizmy
Candida albicans, Candida guilliermondii, Candida tropicalis, Lodderomyces elongisporus , Candida parapsilosis, Clavispora lusitaniae

Tabuľka 3.2: Zoznam druhov, v ktorých boli hľadané ortológy pre telomérické proteíny z kvasinky *Saccharomyces cerevisiae*.

razuje, ktoré z kvasiniek sú evolučne najbližšie ku kvasinke *Saccharomyces cerevisiae*. Dáta (genómy a sekvencie proteínov) pre toto hľadanie sme získali zo stránky BROAD Institute of MIT. Zoznam analyzovaných proteínov je v tabuľke 3.1.

Metódu RBH sme použili pre všetky uvedené proteíny, profilové HMM a sekvenčné motívy len pre proteíny, ktoré mali v databáze KEGG (Kanehisa, 2002) uvedené aspoň 4 ortologické proteíny, z ktorých bolo možné natrénovať HMM a určiť sekvenčné motívy. V databáze KEGG boli dáta o ortologických proteínoch uvedené pre 28 z analyzovaných proteínov. Pre tieto proteíny sme z databázy KEGG zobrali sekvencie ich ortologických proteínov z iných organizmov (príklady pre proteíny CDC13 a EST3 v tabuľke 3.4), pomocou nástroja CLUSTALW (Larkin et al., 2007) sme z tejto skupiny sekvencií vytvorili viacnásobné zarovnanie, ktoré sme následne použili na vytvorenie profilového HMM pomocou nástroja HMMER (Eddy, 1998) (program hmmbuild). Následne sme týmto modelom pomocou nástroja HMMER (program hmmsearch) hľadali ortológ k vybraným proteínom v spomínaných genómoch. Z rovnakých sekvencií sme pomocou nástroja MEME získali zoznam 7 sekvenčných motívov zachovaných v sekvenciách týchto proteínov. Získané motívy sme tiež použili na hľadanie ortológov pomocou programu MAST (Bailey and Gribskov,

Protein	Calb	Ctro	Cpar	Lelo	Cgui	Clus
CDC13	NNN	NNN	NNN	NNN	NNN	NNN
ELG1	AAA	AAA	AAA	AAA	AAA	AAA
EST1	AAA	AAA	NNN	NNN	AAA	NNA
EST3	NAA	NAA	NNN	NNN	NAA	AAA
EXO1	AAA	AAA	AAA	AAA	AAA	AAA
HCM1	AAA	AAA	AAA	AAA	AAA	AAA
HTL1	NNN	NNN	NNN	NNN	NNN	NNN
MCM3	AAA	AAA	AAA	AAA	AAA	AAA
MEC3	AAA	AAA	AAA	AAA	AAA	AAA
MRC1	AAA	AAA	AAA	AAA	AAA	AAA
MRE11	AAA	AAA	AAA	AAA	AAA	AAA
MUS81	AAA	AAA	AAA	AAA	AAA	ANA
NPL6	AAA	AAA	AAA	AAA	AAA	AAA
PRI1	AAA	AAA	AAA	AAA	AAA	AAA
RAD50	AAA	AAA	AAA	AAA	AAA	AAA
RAD51	AAA	AAA	AAA	AAA	AAA	AAA
RAD57	AAA	AAA	AAA	AAA	AAA	ANA
RAD59	AAA	AAA	AAA	AAA	AAA	AAA
RFA1	AAA	AAA	AAA	AAA	AAA	AAA
RFA2	AAA	AAA	AAA	AAA	AAA	AAA
RIF1	AAA	AAA	ANA	NNA	NNA	ANA
RSC58	AAA	AAA	ANN	AAA	AAA	NNN
SIR2	AAA	AAA	AAA	AAA	AAA	AAA
SIR4	NNN	NNN	NNN	NNN	NNN	NNN
STN1	NNN	NNN	NNN	NNN	ANN	NNN
SUA5	AAA	AAA	AAA	AAA	AAA	AAA
SUM1	NNN	NNA	NNA	NNA	NNN	NNN
XRS2	NNN	NNN	NNA	NNN	NNA	NNN

Tabuľka 3.3: Tabuľka s výsledkami hľadania ortológov pre 28 proteínov. Pre každý skúmaný organizmus a proteín boli použité 3 metódy, ktorých výsledky sú uvedené v poradí: RBH, profilové HMM, MAST (hľadanie sekvenčných motívov). Nájdenie signifikantného nálezu pre hľadaný proteín je vyjadrené písmenom A. Písmeno N znamená, že ortológ nebol nájdený.

Proteín	Organizmy
Cdc13	*Ashbya gossypii, Kluyveromyces lactis, Candida glabrata*
Est3	*Ashbya gossypii, Kluyveromyces lactis, Debaryomyces hansenii, Pichia stipitis, Candida glabrata*

Tabuľka 3.4: Zoznam druhov pre tvorbu skupín ortológov pre proteíny CDC13 a EST3. Hľadanie ortológov bolo vykonané podľa proteínov z *S. cerevisiae* v databáze KEGG.

1998). Takýto spôsob hľadania sa sústredí na časti sekvencie, ktoré sú výrazne zachované vo všetkých ortológoch daného proteínu a tak sa dosiahne vyššia senzitivita.

Najskôr programom tblastn (hľadá proteínový dotaz v nukleotidovej databáze) hľadali hity. Pre väčšinu (96 z 121) týchto proteínov boli nájdené signifikantné hity v ostatných genómoch. Ďalšia analýza zahrňovala hľadanie pomocou profilových HMM a sekvenčných motívov vytvorených poďla ortológov v databáze KEGG. Toto hľadanie sme spravili pre 28 proteínov pre ktoré bolo možné vytvoriť profilové HMM a sekvenčné motívy. Výsledky tohto hľadania sú uvedené v tabuľke 3.3.

Z proteínov, ktoré podľa výsledkov hľadania chýbali vo viacerých druhoch, sme následne vybrali niektoré pre ďalšiu analýzu. V ďalšej časti popíšeme zaujímavé výsledky pre vybrané dva proteíny.

Proteín EST3.

Proteín EST3 je jedným z komponentov enzýmu telomerázy, ktorý pôsobí pri predlžovaní telomérických úsekov chromozómov (Lingner et al., 1997; Lendvay et al., 1996). Ďalšie proteíny tohto komplexu, s ktorými sa EST3 interaguje sú EST1, EST2, a RNA komponent TLC1 (Lee et al., 2008; Hughes et al., 2000). EST3 sa na telomerázu viaže pomocou OB-fold domény (Oligosaccharid Binding) (Lee et al., 2008).

Proteíny EST1, EST2 a EST3 sme hľadali vo vyššie uvedených genómoch pomocou programu tblastn. Proteín EST2 mal signifikantný hit (E-value < 0.001) v každom prehľadávanom genóme, EST1 s výnimkou *Candida parapsilosis* a *Lodderomyces elongisporus* tiež, ale EST3 mal výsledok s prijateľnou E-value iba v *Debaryomyces hansenii*. Kvôli tomuto výsledku boli ortológy EST3 (podľa databázy KEGG) použité na vytvorenie HMM

	Candida albicans	Candida guilliermondii	Candida tropicalis	Debaryomyces hansenii	Lodderomyces elongisporus	Candida parapsilosis
tblastn	0.56	2.9	1.9	3e-05	0.78	1.8
HMMER	2.3e-119	6.4e-028	3.4e-052	3e-123	0.0055	0.0016
MAST	3.6e-78	5e-29	9.2e-43	1.1e-83	0.49	0.024

Tabuľka 3.5: E-value pre najlepšie zhodujúce sa úseky genómu nájdené jednotlivými nástrojmi pre proteín Est3.

nástrojom HMMER. Výsledky hľadania týmto modelom boli úspešnejšie, hity boli nájdené pre tie isté druhy ako v prípade EST1 (čiže *Candida albicans, Candida guilliermondii, Candida tropicalis, Debaryomyces hansenii*). Tento výsledok bol následne overený pomocou hľadania motívov vytvorených z rovnakých ortológov pomocou nástroja MAST. Výsledky pre proteín EST3 sú uvedené v tabuľke 3.5. Tieto výsledky oboch hľadaní pre organizmy, v ktorých tblastn nenašiel významné hity, mali veľmi nízku E-value a ukazovali na rovnakú sekvenciu z prehľadávaného genómu. Tieto nájdené sekvencie sa dajú považovať za kandidátov na ortológy EST3 v daných organizmoch a tento predpoklad môže byť podrobený analýze prostredníctvom kritérií, ktoré nevychádzajú zo sekvenčnej podobnosti.

Proteín CDC13.

Esenciálny proteín CDC13 sa viaže pomocou OB-fold domény na jednovláknové prečnievanie telomér a umožňuje tak tvorbu rôznych proteínových komplexov, ktoré buď pozitívne alebo negatívne regulujú replikáciu telomérických úsekov v organizme *Saccharomyces cerevisiae* (Chandra et al., 2001; Meier et al., 2001). Tieto komplexy vznikajú interakciami CDC13 s ďalšími proteínmi. V databáze UNIPROT sú uvedené proteíny (POL1, EST1, FUN12, STM1, STN1 a TEN1), ktoré podľa dostupných štúdií interagujú s CDC13 (Hayashi and Murakami, 2002; Qi and Zakian, 2000; Chandra et al., 2001; Meier et al., 2001).

Pre proteín CDC13 nebol pri hľadaní pomocou tBlastn nájdený v ďalších organizmoch dostatočne signifikantný hit, podobne ani pre proteíny STN1 a TEN1. Naopak zvyšné interagujúce proteíny POL1, EST1, FUN12 a STM1 mali zodpovedajúce hity vo všetkých

	Candida albicans	Candida guilliermondii	Candida tropicalis	Debaryomyces hansenii	Lodderomyces elongisporus	Candida parapsilosis
tblastn	2.5	4.6	*N*	0.85	1.6	3.9
HMMER	*N*	*N*	*N*	*N*	*N*	*N*
MAST	3.3	2.4	0.63	0.069	3	7.9

Tabuľka 3.6: E-value pre najlepšie zhodujúce sa úseky genómu nájdené jednotlivými nástrojmi pre proteín CDC13. *N* znamená, že nástroj nenašiel žiadny úsek, to znamená ani len úsek s malým štatistickým významom.

použitých druhoch. CDC13 zohráva kľúčovú úlohu pre naviazanie telomerázy na teloméry. EST1 sa viaže na CDC13 (Lustig, 2001) a keďže proteín EST3 z telomerázového komplexu bol nájdený pomocou profilových HMM, boli ortológy CDC13 hľadané rovnakým spôsobom (ortológy z KEGG, hľadanie HMMERom a MASTom). V tomto prípade ani jeden z použitých nástrojov nenašiel žiadne signifikantné hity (tabuľka 3.6). Tento výsledok vyvoláva otázku, akým spôsobom sa viaže telomeráza na teloméry v iných organizmoch, ktoré tiež obsahujú telomerázové proteíny EST1, EST2 a EST3. Pre nájdenie proteínu, ktorý viaže telomerázu v týchto organizmoch, by bola vhodná metóda, ktorá zohľadňuje aj iné kritériá ako len sekvenčnú podobnosť.

Zhrnutie

Výsledky štúdie ukazujú, že metódy založené na sekvenčnej podobnosti sú vhodné na automatické identifikovanie ortológov pre veľké množstvo hľadaných proteínov. V určitých prípadoch však tieto metódy nenájdu žiadne ortológy napriek tomu, že ich prítomnosť je predpokladaná na základe dôležitosti biologickej funkcie hľadaného proteínu. Príkladom takýchto proteínov sú napríklad CDC13 a EST3, ktoré obsahujú OB-fold doménu. Táto doména zaujíma v preistore charakteristickú štruktúru a uvedené výsledky naznačujú vhodnosť domény OB-fold ako relevantného kandidáta pre vytvorenie deskriptora charakterizujúceho štrukturálne vlastnosti hľadanej domény. V ďalšej časti preto navrhneme spôsob ako reprezentovať vlastnosti proteínových domén pomocou špecializovaných deskriptorov a popíšeme deskriptor pre OB-fold doménu.

3.3 Deskriptory pre proteínové domény

Na charakterizáciu proteínových domén použijeme reprezentáciu pomocou deskriptorov, ktoré vychádzajú z reprezentácie RNA štrukturálnych motívov. Deskriptory pre proteínové domény popisujú zvolenú doménu podľa charakteristických vlastností jej štruktúry a sekvencie.

Deskriptor pre proteínové domény je vytvorený rozdelením aminokyselinovej sekvencie proteínu na segmenty, ktoré popisujú základné bloky v sekundárnej štruktúre domény. Deskriptor pozostáva z dvoch základných častí. Prvou je postupnosť segmentov pre úseky sekvencie s danou sekundárnou štruktúrou. Druhou časťou je množina definovaných väzieb medzi segmentami v deskriptore. Tieto väzby si v grafickej reprezentácii môžeme predstaviť ako hrany medzi jednotlivými segmentami.

Základom deskriptora teda je postupnosť segmentov, ktoré zodpovedajú úsekom sekvencie s určitou sekundárnou štruktúrou. Tieto segmenty sú identifikované menom, v ktorom je zakódovaný typ sekundárnej štruktúry aminokyselín v danom úseku sekvencie. Prvé písmeno názvu (A,B alebo C) môže indikovať α-helix, β-list alebo žiadnu štruktúru (*coil*) na danom úseku. Príklad definovania segmentov v deskriptore:

$$B1|C1|B2|C2|A1$$

Tento deskriptor obsahuje dva coil segmenty (bez špecifickej sekundárnej štruktúry), dva β-listové a jeden α-helixový segment. Deskriptor špecifikuje v akom poradí búdú tieto úseky nasledovať v sekvencii. Pre každý z týchto segmentov je potrebné definovať minimálnu a maximálnu dĺžku. V prípade, že segment obsahuje dobre zachovaný sekvenčný motív, môžeme preň špecifikovať PSSM maticu (označená identifikátorom B1_LOGO).

$$B1:\ 6\ 15\ B1\_LOGO$$

Skórovacia matica pre tento motív bude reprezentovať pravdepodobnosť výskytu každej aminokyseliny pre každé miesto v motíve - matica má rozmery $20 \times dĺžka\_motívu$. Príklad takejto matice na obrázku 3.4. Motív nemôže byť dlhší ako je maximálna dĺžka segmentu, ktorý ho má obsahovať.

```
B1_LOGO: [3]
-0.7 -0.77 -0.93 0.47 -0.37 -1.14 -0.3 -1.61 0.46 0.16 -0.83 0.19 -1.45 0.24 -0.61 -0.48 0.53 -0.76 1.36 0.61
0.42 1.11 0.66 -0.38 0.38 0.59 0.4 -0.8 -2.03 0.99 0.33 -0.55 -0.77 -1.51 -0.63 -0.98 -1.13 -2.06 -1.43 -1.36
-1.13 -0.66 -0.37 -1.48 -0.39 -1.17 -0.41 -0.78 -1.52 -0.35 0.01 -1.01 -1.45 -0.06 0.73 0.86 0 -0.22 1.16 0.64
```

Obr. 3.4: Príklad PSSM matice pre motív dĺžky 3 v β-listovom segmente B1.

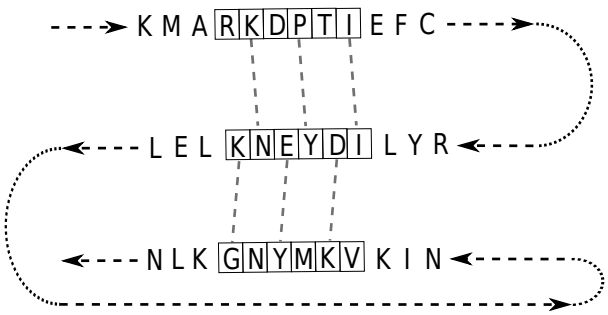
Druhá časť deskriptoru popisuje hydrogénové väzby, ktoré vznikajú medzi β-listami po poskladaní sa v priestore. Tieto väzby reprezentujeme *hranami* medzi dvoma segmentami typu B, pričom každá takáto hrana má určený smer, mená interagujúcich segmentov a dĺžku. Smer hrany (paralelný alebo antiparalelný) je určený podľa vzájomného smeru dvoch interagujúcich β-listov. Rovnaký smer oboch β-listov zodpovedá paralelnej hrane, opačný smer antiparalelnej. Smer má každá hrana špecifikovaný znamienkami + alebo - (pre paralelný respektíve antiparalelný smer). Ďalej sú špecifikované identifikátory dvoch interagujúcich segmentov a nakoniec počet párov interagujúcich aminokyselín, ktoré tvoria popisovanú väzbu medzi segmentami. Počet týchto párov nazývame *dĺžkou hrany*. Príklad zadefinovania hrany medzi segmentami B1 a B2 v antiparalelnom smere, ktorá pozostáva z 3 párov aminokyselín:

$$- \; B1 \; B2 : \; 3$$

Jedna aminokyselina môže vystupovať najviac v jednej väzbe. Medzi dvoma aminokyselinami patriacimi k rovnakej hrane je vždy práve jedna aminokyselina. Toto umožňuje, aby sa na jednom β-liste nachádzali striedavo aminokyseliny patriace do dvoch rôznych hrán a zapadali tak do seba ako zuby v zipse. Schéma takéhoto striedavého zapojenia interagujúcich aminokyselín je na obrázku 3.5.

Takto definovaný deskriptor stanovuje minimálnu dĺžku sekvencie, ktorá obsahuje doménu: súčet minimálnych dĺžok pre všetky segmenty. Okrem tohto obmedzenia nie sú v deskriptore ďalšie striktné podmienky a deskriptor môžeme zarovnať k ľubovoľnej vstupnej sekvencii aminokyselín.

Definícia 3.1. *Zarovnaním* deskriptora ku sekvencii rozumieme určenie začiatočného bodu pre deskriptor na sekvencii, ďalej určenie rozmeru pre každý segment deskriptora, určenie začiatočného bodu pre každý sekvenčný motív špecifikovaný pre segmenty a začiatočný



Obr. 3.5: Striedajúce sa väzby spájajúce tri β-listy. Prvý a druhý sú spojené v antiparalelnom smere, druhý a tretí v antiparalelnom.

pár aminokyselín pre každú hranu.

Charakteristické črty domény (typ sekundárnej štruktúry segmentov, sekvenčné motívy, väzby medzi segmentami) popísané v deskriptore sú využité pri ohodnocovaní daného zarovnania deskriptoru ku sekvencii. Pre charakteristiky zadané v deskriptore potrebujeme skórovaciu schému, ktorá určí ako veľmi vstupná sekvencia vyhovuje očakávaniam vyplývajúcim zo zarovnania deskriptora na sekvenciu. S daným deskriptorom a skórovacou schémou je možné nájsť najlepšie zarovnanie k vstupnej sekvencii a skóre tohto zarovnania použiť na rozhodnutie, či vstupný proteín obsahuje doménu popísanú deskriptorom.

3.3.1 Skórovanie zarovnania deskriptoru ku sekvencii

Deskriptor špecifikuje vlastnosti primárnej, sekundárnej a terciárnej štruktúry popisovanej domény. Skóre zarovnania S je preto určené lineárnou kombináciou týchto troch zložiek.

$$S = a * SecStr + b * SeqMot + c * HydBond$$

, kde a, b a c sú váhy pre jednotlivé zložky skóre. Samotné zložky skóre sa vypočítajú nasledovne:

- **Skóre za sekundárnu štruktúru SecStr** - Táto zložka hodnotí, ako dobre korešpondujú typy sekundárnej štruktúry segmentov s predikovanou štruktúrou vstupného proteínu na mieste, kde bol zarovnaný deskriptor.

Každá aminokyselina sekvencie a_i môže patriť k jednému z troch typov sekundárnej štruktúry (α-helix ,β-list alebo $coil$). Nech $p_{a_i}(t)$ je pravdepodobnosť, že aminokyselina a_i má sekundárnu štruktúru typu t ($t \in \{\alpha\text{-}helix, \beta\text{-}list, coil\}$. Nech segment j predstavuje úsek sekvencie so sekundárnou štruktúrou typu u a nech je zarovnaný k vstupnej sekvencii od aminokyseliny a_k po a_l. Potom skóre tohto segmentu s_j za sekundárnu štruktúru spočítame ako sumu zlogaritmovaných pravdepodobností p_{a_i} pre zodpovedajúci typ štruktúry u pre každú aminokyselinu a_i v intervale $< a_k, a_l >$. Skóre pre segment j sa teda vypočíta nasledovne: $s_j = \sum_{i=k}^{l} \ln(p_{a_i}(u) + 0.5)$

Ku každej pravdepodobnosti sa pripočíta 0.5, čím dosiahneme, že pozitívne skóre budú mať len aminokyseliny, ktorých pravdepodobnosť príslušnej štruktúry je väčšia ako 50 %. Takýmto skórovaním preferujeme zarovnania mapujúce segmenty na vhodné aminokyseliny a zároveň zabezpečujeme určitú toleranciu k chybám, ktoré nevyhnutne nastávajú pri predikcii sekundárnej štruktúry. Celkové skóre zarovnaného deskriptora za sekundárnu štruktúru SecStr dostaneme súčtom skóre pre všetky segmenty s_j v deskriptore.

$$SecStr = \sum s_j$$

Pre spočitanie tohto skóre potrebujeme mať pre každú aminokyselinu v sekvencii určenú pravdepodobnosť p_{a_i} pre príslušnosť aminokyseliny k jednotlivým typom sekundárnej štruktúry. V našej práci sme použili aposteriórne pravdepodobnosti získané predikciou pomocou nástroja PSIPRED (Jones, 1999). Tento nástroj najskôr zo vstupnej sekvencie vytvorí sekvenčný profil porovnaním s uloženou databázou sekvencií. Takto vytvorený profil je rozdelený na okná dĺžky 15, ktoré sú použité ako vstup do natrénovanej neurónovej siete. Použitie sekvenčného profilu namiesto len vstupnej sekvencie zlepšuje predikčnú schopnosť tejto siete. Neurónová sieť má 3 výstupy, ktoré predstavujú pravdepodobnosti jednotlivých typov sekundárnej štruktúry pre aminokyselinu v strede okna, ktoré bolo na vstupe neurónovej siete. Pri počítaní skóre zarovnania musia byť tieto pravdepodobnosti súčasťou vstupu spolu so samotnou sekvenciou, ku ktorej je deskriptor zarovnávaný.

- **Skóre za sekvenčné motívy SeqMot** - Táto zložka skóre vyjadruje zhodu medzi vstupnou sekvenciou a sekvenčnými motívmi špecifikovanými pre niektoré segmenty. Tieto motívy predstavujú podmienky na očakávanú primárnu štruktúru sekvencie, ktorá by mala obsahovať doménu popísanú deskriptorom.

  Nech segment j je zarovnaný k vstupnej sekvencii od aminokyseliny k po l a obsahuje motív m_j, ktorý má dĺžku $len(m_j)$. Motív je reprezentovaný PSSM maticou, ktorá je súčasťou deskriptora. PSSM matica má rozsah $20 \times len(m_j)$ a pre každú aminokyselinu určuje pravdepodobnosť výskytu na každej pozícii v motíve. Nech $PSSM_{m_j}(a, i)$ vyjadruje pravdopodobnosť výskytu aminokyseliny a na pozícii i v motíve m_j. Ak motív m_j začína v zarovnaní na aminokyseline $n (n \in < k, l - len(m_j) >$), tak pre každú aminokyselinu $a_i (i \in < n, n + len(m_j) >)$ na úseku so zarovnaným motívom túto pravdepodobnosť určime ako $PSSM_{m_j}(a_i, i - n)$. Pravdepodobnosti v matici sú zlogaritmované, čo umožňuje spočítať skóre celého motívu ako sumu prvkov tejto matice vybraných podľa zodpovedajúcich aminokyselín na jednotlivých pozíciách motívu.

  Takže skóre za motív m_j, ktorý v zarovnanej sekvencii začína na aminokyseline a_n, vypočítame sumou $\sum_{l=0}^{len(m_j)} PSSM_{m_j}(a_{n+l}, l)$. Celkové skóre za sekvenčné motívy SeqMot dostaneme súčtom skóre pre všetky motívy v deskriptore.

$$SeqMot = \sum_{m_j} \sum_{l=0}^{len(m_j)} PSSM_{m_j}(a_{n+l}, l)$$

- **Skóre za hydrogénové väzby HydBond** - Táto zložka skóre hodnotí kompatibilitu segmentov vystupujúcich v hranách deskriptora vytvoriť hydrogénové väzby, ktoré sa po poskladaní vytvoria v doméne popisovanej deskriptorom.

  Každá hrana predstavuje určitý počet spárovaných aminokyselín. Ak je medzi segmentami a a b hrana dĺžky $len(h_{a,b})$, zarovnanie určí v oboch segmentoch začínajúci pár hrany $i_{h_{a,b}}$ a $k_{h_{a,b}}$. Skóre za hranu získame ako súčet skór $(bond(i_{h_{a,b}}, k_{h_{a,b}}))$ pre jednotlivé páry v hrane: $h_{ab} = \sum_{l=0}^{len(h_{a,b})} bond(i_{h_{a,b}} + 2l, k_{h_{a,b}} + 2l)$. V prípade antiparalelného smeru hrany sa počítajú dvojice $bond(i_{h_{a,b}} + 2l, k_{h_{a,b}} - 2l)$. Skóre za

hydrogénové väzby HydBond dostaneme súčtom skóre pre všetky hrany v deskriptore.

$$HydBond = \sum_{h_{a,b}} \sum_{l=0}^{len(h_{a,b})} bond(i_{h_{a,b}} + 2l, k_{h_{a,b}} + 2l)$$

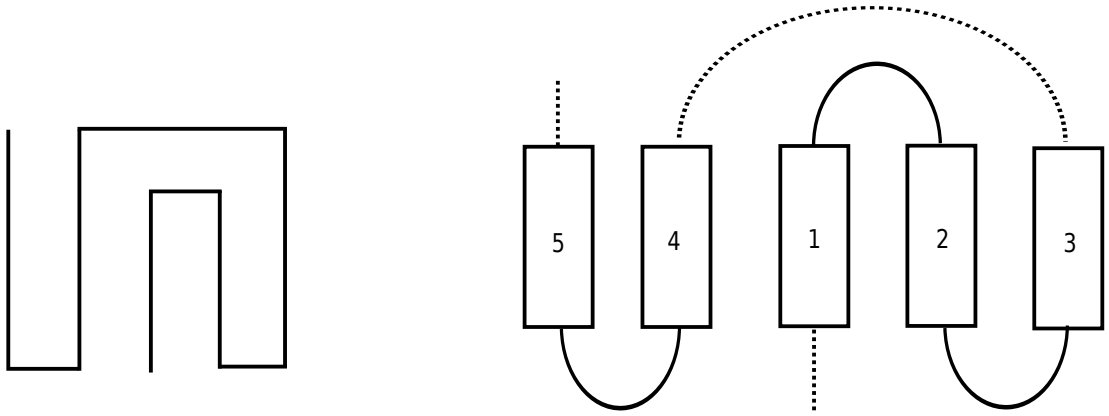
Rozhodnúť či dve vzdialené aminokyseliny vytvoria hydrogénovú väzbu je problém, ktorý sme skúsili vyriešiť metódami strojového učenia. Na ohodnotenie samotného páru v hrane sme natrénovali klasifikátor na základe *support vector machine*, ktorý detailne popíšeme v kapitole 4.

Výsledné skóre pre dané zarovnanie deskriptoru na sekvenciu dostaneme lineárnou kombináciou skór pre jednotlivé zložky s danými váhami. Tieto váhy môžu byť určené *ad hoc*, prípadne natrénované na množine pozitívnych a negatívnych príkladov hľadanej domény. V ďalšej časti uvedieme deskriptor pre Telo_bind doménu, ktorú obsahujú niektoré telomérické proteíny analyzované v prípadovej štúdii.

3.3.2 Deskriptor pre Telo_bind doménu

Vyššie popísaný typ deskriptoru sme vytvorili pre *Telo_bind* doménu z rodiny OB-foldových domén, do ktorej patria proteíny CDC13 a EST3 z prípadovej štúdie pre telomérické proteíny. Táto doména, podobne ako ostatné z rodiny OB-fold, sa skladá z piatych β-listov sformovaných do β-barelu, ktorý je zväčša uzavretý α-helixom (Arcus, 2002) a β-listy v barele sú zapojené do motívu *gréckeho kľúča*: jednotlivé listy sú zapojené v poradí (1-2-3 5-4-1) (Theobald et al., 2003). Schéma tohto motívu je na obrázku 3.6.

Základ deskriptora tvoria segmenty zodpovedajúce týmito β-listami a α-helixom, ktoré sú prekladané coilovými segmentami. Na obrázku 3.7 je zobrazená schéma deskriptora pre Telo_bind doménu, ktorý má 5 β segmentov, jeden α segment a 4 coil segmenty. Ohraničenia na minimálnu a maximálnu dĺžku jednotlivých segmentov boli určené na základe zarovnania 15 OB-fold domén z rôznych proteínov uvedené v článku (Theobald et al., 2003). Sekvenčné motívy boli na základe HMM modelu pre rodinu Telo_bind z databázy Pfam (Finn et al., 2007) definované pre 6 segmentov (4 β-listy a 2 coil segmenty), ktoré obsahovali úseky s veľkou zhodou medzi doménami v zarovnaní. Na obrázku 3.7 sú tieto sekvenčné

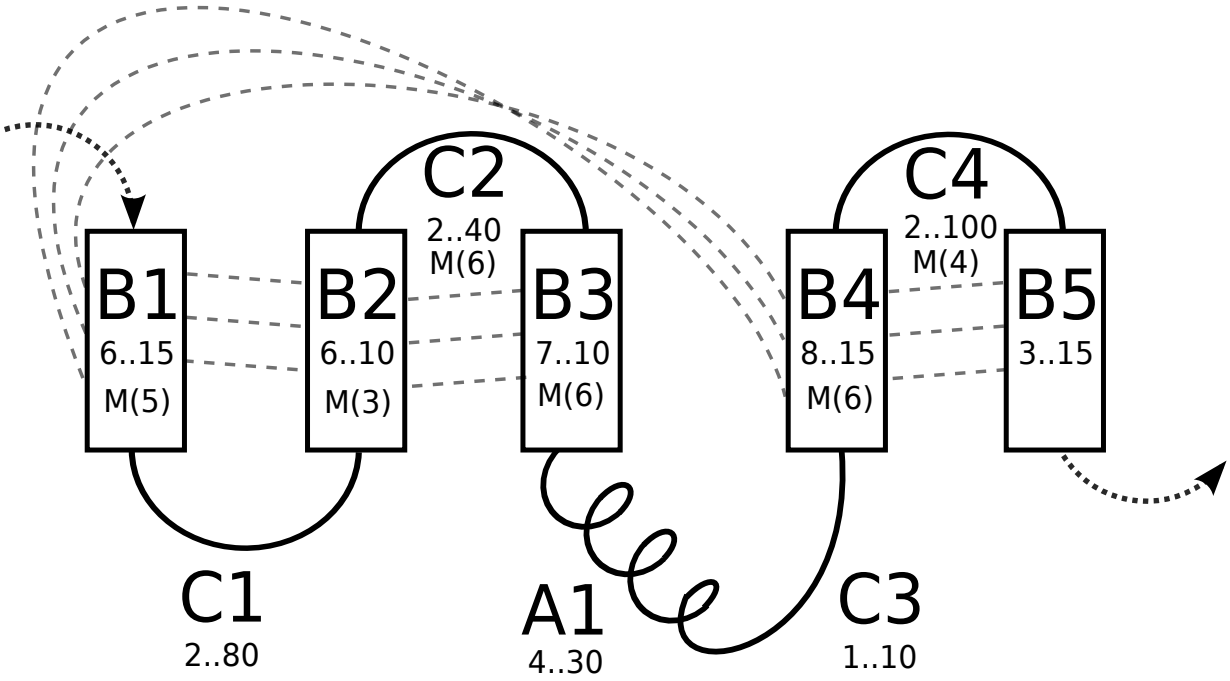


Obr. 3.6: Motív gréckeho kľúča, do ktorého sa viažu β-listy v OB-fold doméne.

motívy označené písmenom M a dĺžkou tohto sekvenčného motívu. PSSM matica týchto sekvenčných motívov bola zostavená z pravdepodobností aminokyselín na daných úsekoch v HMM. Väzby a ich dĺžky boli odvodené na základe analýzy 3D modelu Telo_bind domény v proteíne CDC13 z kvasinky *Saccharomyces Cerevisiae* uloženého v databáze PDB (Rose et al., 2011). Väzby medzi jednotlivými segmentami sú v schéme zobrazené šedými prerušovanými čiarami.

Kompletný popis deskriptoru je na obrázku 3.8. Deskriptor najprv špecifikuje počet, typ a poradie segmentov v popisovanej doméne. Následne je pre každý segment uvedená minimálna a maximálna dĺžka. Ak segment obsahuje sekvenčný motív, je tomuto motívu priradený identifikátor (napríklad B1_LOGO). Po definovaní segmentov sú v deskriptore určene väzby medzi jednotlivými segmentami. V poslednej časti deskriptoru sú vypísané PSSM matice pre sekvenčné motívy segmentov. Pri identifikátore sekvenčného motívu je zadaná jeho dĺžka, pričom motív je určený zodpovedajúcim počtom riadkov s 20 číslami, ktoré vyjadrujú pre každú z dvadsiatich aminokyselín zlogaritmovanú pravdepodobnosť výskytu danej aminokyseliny na danom mieste v sekvenčnom motíve.

Deskriptor vytvorený pre doménu Telo_bind na obrázku 3.8 popisuje jej charakteristické črty v primárnej (sekvenčné motívy), sekundárnej (typy segmentov) aj terciárnej štruktúre (väzby medzi segmentami). Ak chceme určiť či daná proteínová sekvencia obsahuje túto doménu, musíme nájsť najlepšie miesto zarovnania deskriptoru ku sekvencii.



Obr. 3.7: Nákres deskriptoru pre OB-fold doménu.

Pre skompletovanie skórovacej schémy potrebujme spôsob ako ohodnotiť páry interagujúcich aminokyselín. Hodnotenie týchto párov popíšeme v 4. kapitole a následne v kapitolách 5 a 6 uvedieme algoritmy, ktorými sa dá určiť najlepšie zarovnanie pre daný deskriptor a vstupnú proteínovú sekvenciu.

```
B1|C1|B2|C2|B3|A1|C3|B4|C4|B5
B1: 6 15 B1_LOGO
C1: 2 80
B2: 6 10 B2_LOGO
C2: 2 40 C2_LOGO
B3: 7 10 B3_LOGO
A1: 4 30
C3: 1 10
B4: 8 15 B4_LOGO
C4: 2 100 C4_LOGO
B5: 3 15
- B1 B2: 3
- B2 B3: 3
- B4 B5: 3
- B1 B4: 3
**********LOGO_DEFINITIONS*************
B1_LOGO: [5]
1.4737 0.5703 1.4760 1.1420 1.6928 -0.2891 0.3289 -2.1094 -0.6210 0.0593 -1.4480 -0.5506 -0.7565 -1.9646 -1.7750 -1.1258 -1.4303 -2.5349 -1.8037 -1.5904
1.5272 0.8311 1.2353 1.2990 1.1484 0.2098 -0.6558 -2.1109 -2.1399 -0.7990 -1.4497 1.6771 -0.7529 -1.9715 -1.7785 -1.1258 -1.4327 -1.4815 -1.8088 -1.5925
-0.6192 -0.2024 0.5459 -1.0728 -0.2116 -0.8002 0.6769 1.2458 -1.9095 -0.8191 -0.3585 -1.0407 -1.3340 -0.8034 -1.0573 -0.8674 1.0839 -1.2990 -0.8317 -1.0114
1.1771 0.8718 2.0237 -0.0035 1.6023 -0.4086 -0.0114 -2.2651 -2.2817 -0.9310 -1.6087 0.3996 -0.9004 -2.1166 -1.9329 -1.2897 -1.5813 -2.6816 -1.9610 -1.7470
1.0375 0.8607 1.8582 1.3712 0.2444 -0.4855 -0.2351 -2.3895 -2.4012 0.6099 -1.7393 -0.8425 -1.0488 -2.2275 -2.0553 -1.4366 -1.7167 -2.7808 -2.0836 -1.8819
B2_LOGO: [3]
-0.7001 -0.7778 -0.9399 0.4729 -0.3775 -1.1448 -0.3035 -1.6139 0.4636 -0.1663 -0.8363 0.1970 -1.4514 -0.2498 -0.6121 -0.4862 0.5370 -0.7639 1.3670 0.6199
0.4238 1.1190 0.6647 -0.3823 0.3862 0.5934 0.4044 -0.8099 -2.0300 0.9983 0.3361 -0.5553 -0.7738 -1.5145 -0.6345 -0.9897 -1.1363 -2.0602 -1.4347 -1.3661
-1.1356 -0.6612 -0.3739 -1.4812 -0.3958 -1.1792 -0.4097 -0.7886 -1.5261 -0.3573 -0.0071 -1.0106 -1.4502 -0.0593 0.7327 0.8667 0.0275 -0.2236 1.1620 0.6415
C2_LOGO: [6]
-1.2004 -1.2746 -1.0686 -1.5443 -0.4577 -1.2408 -0.4658 0.3925 -1.5785 1.0427 0.2566 -1.0691 -1.5109 0.1226 -0.0346 -0.3857 -0.0263 0.0303 0.5472 0.2239
-1.2241 -1.2970 -1.0913 -1.5670 -0.4810 -1.2636 0.2913 -1.2015 -1.5925 -0.6282 -0.0552 -1.0890 -1.5326 0.1429 -0.4914 -0.4013 0.7431 1.5597 0.2294 0.3209
0.4298 -0.0008 0.1128 1.7583 0.7730 0.4148 -0.5505 -1.9901 -2.0220 0.7997 -0.2804 1.8756 1.6642 -1.7735 -1.6235 -0.0245 -1.2723 -2.3375 -1.6307 -1.4477
1.0995 -0.1109 0.7143 1.4341 1.2731 1.9024 -0.5386 -1.9787 -2.0111 1.0155 -0.3144 -0.0074 -0.6499 -1.7555 -1.6118 -0.9876 -1.2590 -2.3206 -0.0279 -1.4347
0.8348 0.5127 0.4136 0.9138 1.2471 2.1332 -0.0201 -1.9592 -1.9930 0.3991 -0.1945 1.1536 -0.6684 -0.6951 -1.5452 -0.9602 -0.5470 -2.1833 -0.3299 -1.3797
-0.5960 -0.6704 -0.1568 -0.5139 0.4815 -0.8514 -0.2966 -1.6151 -1.6712 1.3435 0.4973 -0.0469 -1.2097 -0.3395 0.8846 -0.5061 0.0190 -0.8545 -0.1108 -0.6127
B3_LOGO: [6]
1.4816 1.4929 0.9900 1.1481 0.8120 -0.3061 0.3342 -2.1468 -2.1625 -0.8224 -1.4886 0.1753 -0.7694 -1.9982 -1.8141 -1.1630 -1.4538 -2.5668 -1.8400 -1.6225
0.6896 0.0312 -0.9218 -1.3763 -0.3485 0.4892 -0.4662 -1.5411 -1.1549 0.7629 -0.1595 0.2524 -1.4192 0.1908 -0.0836 0.4550 0.2400 -0.6772 0.5539 0.5435
1.2712 1.2755 1.8097 0.0850 0.2756 1.6395 0.2403 -2.3971 -2.4010 -1.0083 -1.7524 -0.8560 -1.0635 -2.2322 -2.0649 -1.4602 -1.7318 -2.7778 -2.0951 -1.8973
0.9279 -0.1828 1.0065 -0.2599 1.0463 -0.6937 -0.1900 -1.6722 -1.7250 -0.0575 0.3810 -0.7732 -1.0810 -0.3308 0.1439 0.1605 0.0963 -1.0055 0.4063 -0.4462
1.2948 0.9465 0.5770 1.0428 1.6543 -0.1852 -0.5444 -1.9931 -2.0243 -0.6875 -0.2484 1.0595 -0.6411 -1.8332 -1.6518 1.5720 -1.3049 -2.4033 -1.6754 -1.4682
-1.7801 -1.5373 -0.9957 2.6519 -1.3476 -2.0056 -2.8088 -3.5837 -3.5055 -2.9132 -3.0167 2.7864 -0.1254 -3.5547 -2.6259 -0.9108 -0.4535 -3.5718 -3.367 -2.9416
B4_LOGO: [6]
-1.4635 -1.5362 -1.3361 -1.8102 -0.7358 -1.5018 -0.7337 1.4460 -0.8681 -0.8811 -0.6106 -1.3305 -1.7714 -0.2975 -0.6765 -0.2688 -0.2728 0.1255 0.9868 -0.3351
0.9047 -1.3111 -1.1013 -1.5618 0.5632 -1.2918 -0.0881 -1.6351 -1.7076 -0.7355 -0.8733 -1.1560 -1.5811 1.1183 -0.6158 0.1436 0.1499 1.5116 -0.3294 -0.5990
2.1667 0.6046 2.0298 -0.9960 0.6780 -0.7965 -0.6313 -2.9289 -1.1343 -1.4157 -2.3139 -1.4319 -1.6495 -2.7189 -2.5953 -2.0848 -2.2923 -3.2227 -2.6259 -2.4652
1.7791 0.2841 0.9555 1.1471 0.4225 1.9092 -0.6239 -2.0595 -2.0921 -0.7671 -0.5847 -0.5328 -0.7424 -1.8012 -1.6779 -1.0662 -0.0022 -2.3636 0.4815 -1.5039
-0.2866 0.6807 -0.2796 -0.3928 -0.1142 -0.1289 0.8489 -1.6828 -1.7380 -0.6778 -0.2659 -0.8910 -1.2239 -0.4310 -0.2793 -0.5774 -0.0575 -0.9468 0.3942 1.2323
1.2308 1.2335 1.0623 0.9988 1.1805 -0.3197 -0.6736 -2.1211 -2.1446 -0.8160 -1.4569 -0.5739 -0.7747 -1.9015 -1.7576 -1.1299 -0.5205 -2.4666 0.8671 -1.5734
C4_LOGO: [4]
-0.6098 -1.3616 -1.4980 2.0622 0.1747 -1.6324 -2.1948 -3.2997 -3.2646 -2.3198 -2.6857 2.9909 1.1494 -3.1681 -2.5387 0.7690 -2.3560 -3.4135 -1.0959 -2.6935
-1.3028 -1.3751 -1.1692 -1.6453 -0.5577 -1.3411 -0.5616 -1.2315 -1.6719 -0.1264 -0.5459 -1.1666 -1.6101 -0.0192 1.2885 -0.1022 1.0699 1.1840 0.2550 -0.1963
-1.2866 -1.3591 -1.1536 -1.6302 -0.5412 -1.3251 -0.5484 0.7090 -0.5820 -0.6905 0.0412 -1.1524 -1.5938 -0.1923 -0.4157 0.6290 0.4628 0.4518 0.5175 0.8348
-1.2794 -1.3525 -0.0907 -1.6231 -0.5342 -1.3183 -0.0667 -1.1158 -1.6574 -0.6842 -0.4241 -1.1463 -1.5877 0.2505 -0.0379 -0.4631 0.7588 -0.3453 1.5243 0.5929
```

Obr. 3.8: Deskriptor pre Telo_bind doménu z rodiny OB-fold domén.

Kapitola 4

Rozpoznávanie aminokyselín spojených hydrogénovou väzbou pomocou SVM

Zo spôsobu skórovania zarovnania deskriptora ku sekvencii, ktorý sme popísali v predchádzajúcej kapitole je zrejmé, že prínos zložky skóre za hydrogénové väzby závisí od kvality metódy, ktorá ohodnocuje spárované aminokyseliny. Tieto páry vznikajú pri skladaní sekvencie proteínu do trojrozmernej štruktúry, takže interakcie, ktoré popisujeme deskriptorom, nastávajú medzi vzdialenými aminokyselinami v rôznych β-listoch. Interakcie predstavujú štatisticky významné závislosti medzi vzdialenými úsekmi sekvencie (Steward and Thornton, 2002) a prihliadanie k tejto informácii môže napríklad pomôcť korigovať predpovedanú štruktúru (Olmea et al., 1999). V našom prípade identifikovanie aminokyselín, ktoré majú tendenciu tvoriť interagujúci pár, pomôže správne určiť β-listy, ktoré tvoria štrukturálnu doménu v proteíne.

Štruktúra je určovaná v pozorovaných proteínoch pomocou experimentov (napríklad kryštalografia, magnetická rezonancia), ktoré určujú *terciárnu* štruktúru, teda súradnice atómov aminokyselín proteínu v priestore. Hydrogénové väzby medzi aminokyselinami sa určia na základe nameraných vzdialeností medzi aminokyselinami v poskladanej štruktúre. Dáta z týchto experimentov sú uložené vo verejne prístupných databázach. Kvôli materiálnej a časovej náročnosti nie je možné určiť štruktúru pre každý z tísícov existujúcich proteínov experimentálne. Pomocou rôznych informatických metód môžeme analyzovať dostupné dáta z experimentov a použiť ich na predikciu štruktúry. V tejto kapitole popíšeme

zostrojenie SVM klasifikátorov, ktoré ohodnocujú zhodu v štruktúre hydrogénových väzieb v zarovnaní deskriptoru ku sekvencii, pomocou metód strojového učenia.

4.1 Páry interagujúcich aminokyselín v β-listoch

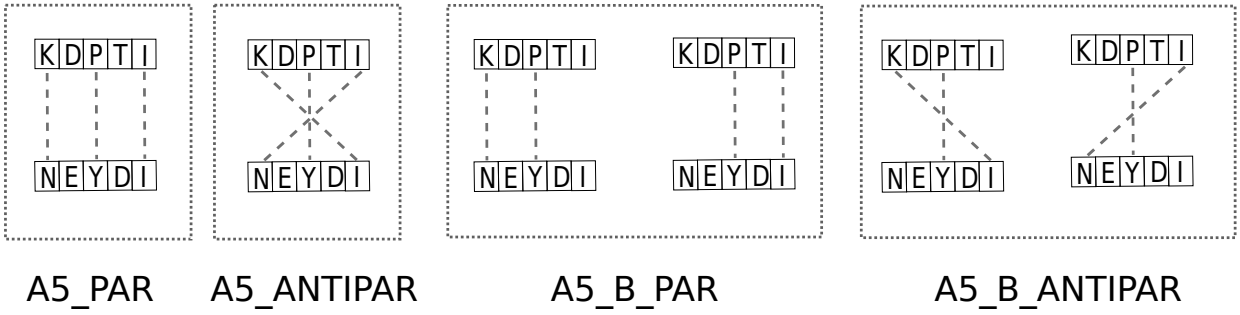
Pre účely našej deskriptorovej metódy je potrebné vytvoriť klasifikátor, ktorý ohodnotí ľubovoľnú dvojicu aminokyselín v proteíne z hľadiska očakávanej tendencie vytvoriť väzbu. Pre naše analýzy sme použili údaje o štruktúre 101642 proteínov z databázy Protein Data Bank (Rose et al., 2011). Z tejto sme pomocou nástroja CD-Hit (Li and Godzik, 2006) vytvorili zhluky sekvencií s 90 % podobnosťou a vybrali z každého zhluku jednu sekvenciu, na ktorej sme pomocou nástroja Jmol Viewer (Jmol, 2012) vypočítali hydrogénové väzby medzi aminokyselinami v týchto sekvenciách.

V 27709 vybraných sekvenciách reprezentujúcich zhluky sme hľadali trojice spárovaných aminokyselín v β-listoch, ktoré nasledovali v sekvencii striedavo za sebou (zipsový vzor). Takéto trojice párov sa dajú reprezentovať párom úsekov sekvencie dĺžky 5, kde vo väzbách vystupujú prvá, tretia a piata aminokyselina v úseku.

Dve pätice aminokyselín pre nás teda predstavujú vzorku. Ak interagujú aminokyseliny v týchto dvoch úsekoch v poradí ($1 \sim 1, 3 \sim 3, 5 \sim 5$), označíme takýto prípad za pozitívnu vzorku pre väzbu v *paralelnom smere*. Ak interagujú aminokyseliny v týchto dvoch úsekoch v poradí ($1 \sim 5, 3 \sim 3, 5 \sim 1$), označíme takýto prípad za pozitívnu vzorku pre väzbu v *antiparalelnom smere*.

Za negatívnu vzorku považujeme dve pätice aminokyselín z β-listov, medzi ktorými nie je žiadna väzba. V spomínaných sekvenciách z databázy PDB sme identifikovali 96755 pozitívnych príkladov a 105914 negatívnych príkladov.

Okrem príkladov, kde v dvoch päticiach interagujú tri páry aminokyselín, sme hľadali aj prípady pätíc aminokyselín, kde interagujú prostredné aminokyseliny ($3 \sim 3$) a len jeden ďalší aminokyselinový pár. Pre paralelný smer sú to väzby ($1 \sim 1$ alebo $5 \sim 5$), pre antiparalelný ($1 \sim 5$ alebo $5 \sim 1$). Takéto prípady sa vyskytujú na začiatku a na konci série spárovaných aminokyselín, ktoré spájajú dva vzdialené β-listy. Takýchto pozitívnych



Obr. 4.1: Zobrazené páry pätíc s väzbami.

trénovacia sada	P	N	testovacia sada	P	N
A3 PAR	70000	350000	A3 PAR TEST	10000	50000
A3 ANTIPAR	70000	350000	A3 ANTIPAR TEST	10000	50000
A5 PAR	70000	350000	A5 PAR TEST	10000	50000
A5 ANTIPAR	70000	350000	A5 ANTIPAR TEST	10000	50000
A5 B PAR	70000	350000	A5 B PAR TEST	10000	50000
A5 B ANTIPAR	70000	350000	A5 B ANTIPAR TEST	10000	50000
A5 MIX	140000	700000	A5 B ANTIPAR TEST	20000	100000

Tabuľka 4.1: Prehľad trénovacích a testovacích množín. Stĺpce P označujú počet pozitívnych príkladov v danej sade. Stĺpce N počet negatívnych príkladov v sade.

paralelných krajných príkladov bolo 146001, antiparalelných 300133. Náčrt trénovacích vstupov je na obrázku 4.1.

Z týchto vzoriek interagujúcich aminokyselín sme vybrali trénovacie sady so 70000 pozitívnymi príkladmi, ku ktorým bol pridaný päťnásobok náhodne vybraných negatívnych príkladov. Teda 350000 párov dvoch úsekov s piatimi aminokyselinami z dvoch β-listov, medzi ktorými nie je žiadna väzba. Takýmto spôsobom sme vytvorili 2 trénovacie sady pre paralelné a antiparalelné príklady s troma väzbami. Sadu pre tieto paralelné príklady označíme A5 PAR a pre antiparalelné A5 ANTIPAR.

Ďalšie dve rovnako veľké trénovacie sady sme vytvorili aj pre okrajové príklady, s jednou väzbou medzi prostrednými aminokyselinami v päticiach a jednou ďalšou väzbou medzi aminokyselinami na okrajoch vzorky. Sadu s takýmito paralelnými príkladmi označíme A5 PAR B a sadu s antiparalelnými A5 ANTIPAR B.

Ku každej trénovacej sade bola vygenerovaná testovacia sada s 10000 pozitívnymi príkl-

testovacia sada	zloženie	P	N
A5 UNION TEST	A5 ANTIPAR TEST + A5 PAR TEST + A5 B ANTIPAR TEST + A5 B PAR TEST	40000	200000

Tabuľka 4.2: Popis zloženia množiny A5 UNION TEST. Stĺpec P označuje počet pozitívnych príkladov v danej sade. Stĺpec N počet negatívnych príkladov v sade.

ladmi a päťnásobkom negatívnych príkladov. Testovacie množiny označíme príponou TEST a teda dostaneme sady A3 PAR TEST, A3 ANTIPAR TEST, A5 PAR TEST, A5 ANTIPAR TEST, A5 B PAR TEST, A5 B ANTIPAR TEST.

Zjednotením testovacích množín A5 PAR TEST a A5 ANTIPAR TEST získame testovaciu množinu A5 MIX TEST, ktorá teda obsahuje 20000 pozitívnych príkladov (10000 paralelných a 10000 antiparalelných) a k nim päťnásobok negatívnych príkladov. Prehľad všetkých vytvorených trénovacích a testovacích množín je uvedený v tabuľke 4.1.

Nakoniec zjednotením množín A5 PAR TEST, A5 ANTIPAR TEST, A5 B PAR TEST a A5 B ANTIPAR TEST sme vytvorili najväčšiu testovaciu množinu A5 UNION TEST, ktorá má 40000 pozitívnych príkladov (10000 paralelných, 10000 antiparalelných, 10000 paralelných okrajových a 10000 antiparalelných okrajových) a päťnásobok negatívnych príkladov (viď. tabuľka 4.2).

Takto vygenerované trénovacie sady sme použili na vytvorenie skórovacej matice a na natrénovanie rôznych klasifikátorov (SVM) pre ohodnotenie potenciálne spárovaných aminokyselín, ktoré popíšeme v ďalšej časti.

4.2 Skórovacie matice

Prvým krokom k vytvoreniu klasifikátora pre väzby bolo vytvorenie skórovacej matice podobnej matici pre zarovnania z kapitoly 2. Táto matica je symetrická a má rozmery 20×20, jeden riadok a jeden stĺpec pre každý druh aminokyseliny. Prvky tejto matice vyjadrujú očakávanie, že dané dve aminokyseliny (prvá určená stĺpcom, druhá určená riadkom) vytvárajú spolu hydrogénovú väzbu, ktorá spája dva vzdialené β-listy. Skóre pre

-0,338 0,532 0,386 0,526 0,463 -0,123 0,205 -0,008 -1,508 0,128 -0,129 0,368 0,263 0,086 0,101 0,189 -0,076 -0,257 -0,097 -0,087
0,532 -0,465 0,479 0,373 0,370 0,115 0,205 -0,284 -1,431 -0,016 -0,306 0,473 -0,137 0,116 -0,334 0,194 -0,132 -0,303 -0,152 -0,074
0,386 0,479 -0,613 0,520 0,372 0,001 0,367 -0,081 -1,563 -0,026 -0,161 0,372 0,183 0,005 -0,388 0,334 -0,129 -0,336 -0,060 0,024
0,526 0,373 0,520 -0,171 0,195 0,407 0,115 0,409 -1,722 0,059 0,159 0,692 0,208 -0,132 -0,442 0,113 -0,209 -0,338 -0,398 0,036
0,463 0,370 0,372 0,195 0,242 0,753 0,122 -0,321 -1,600 0,059 -0,479 0,209 0,578 -0,156 -0,046 -0,056 0,284 -0,240 -0,119 -0,319
-0,123 0,115 0,001 0,407 0,753 -0,758 0,266 -0,502 -2,992 -0,483 1,015 0,036 -0,387 -0,675 -0,383 0,490 -0,337 -0,682 0,793 -0,292
0,205 0,205 0,367 0,115 0,122 0,266 -0,778 -0,210 -2,008 -0,414 -0,539 0,086 0,006 -0,619 -0,614 0,013 -0,148 -0,399 -0,102 -0,178
-0,008 -0,284 -0,081 0,409 -0,321 -0,502 -0,210 -1,327 -2,018 -0,569 -0,708 0,275 0,175 -0,734 -0,430 -0,562 -0,520 -0,615 -0,939 -0,472
-1,508 -1,431 -1,563 -1,722 -1,600 -2,992 -2,008 -2,018 -2,240 -1,686 -1,819 -1,933 -2,324 -0,891 -1,257 -1,319 -1,354 -1,640 -1,209 -0,858
0,128 -0,016 -0,026 0,059 0,059 -0,483 -0,414 -0,569 -1,686 -0,902 0,259 -0,378 -0,383 0,119 -0,184 0,034 0,285 -0,029 0,215 0,195
-0,129 -0,306 -0,161 0,159 -0,479 1,015 -0,539 -0,708 -1,819 0,259 -0,430 -0,250 -0,144 -0,293 0,064 -0,064 -0,013 -0,432 -0,008 -0,069
0,368 0,473 0,372 0,692 0,209 0,036 0,086 0,275 -1,933 -0,378 -0,250 0,546 0,568 0,106 -0,269 0,050 0,101 -0,448 -0,019 0,222
0,263 -0,137 0,183 0,208 0,578 -0,387 0,006 0,175 -2,324 -0,383 -0,144 0,568 -1,382 -0,303 -0,057 1,398 0,350 -0,705 -0,148 -0,008
0,086 0,116 0,005 -0,132 -0,156 -0,675 -0,619 -0,734 -0,891 0,119 -0,293 0,106 -0,303 -1,127 0,152 0,239 0,296 -0,393 1,215 1,158
0,101 -0,334 -0,388 -0,442 -0,046 -0,383 -0,614 -0,430 -1,257 -0,184 0,064 -0,269 -0,057 0,152 -0,835 -0,499 0,027 -0,220 -0,193 -0,345
0,189 0,194 0,334 0,113 -0,056 0,490 0,013 -0,562 -1,319 0,034 -0,064 0,050 1,398 0,239 -0,499 -0,328 -0,298 -0,239 -0,254 -0,122
-0,076 -0,132 -0,129 -0,209 0,284 -0,337 -0,148 -0,520 -1,354 0,285 -0,013 0,101 0,350 0,296 0,027 -0,298 -0,539 -0,198 0,414 0,616
-0,257 -0,303 -0,336 -0,338 -0,240 -0,682 -0,399 -0,615 -1,640 -0,029 -0,432 -0,448 -0,705 -0,393 -0,220 -0,239 -0,198 -1,482 0,362 0,268
-0,097 -0,152 -0,060 -0,398 -0,119 0,793 -0,102 -0,939 -1,209 0,215 -0,008 -0,019 -0,148 1,215 -0,193 -0,254 0,414 0,362 -0,587 -0,177
-0,087 -0,074 0,024 0,036 -0,319 -0,292 -0,178 -0,472 -0,858 0,195 -0,069 0,222 -0,008 1,158 -0,345 -0,122 0,616 0,268 -0,177 -0,666

Obr. 4.2: Príklad skórovacej matica pre väzby v β-listoch.

jednotlivé prvky matice sa vypočitajú nasledovne:

$$S_{ij} = \log\left(\frac{b_{ij}}{f_i \cdot f_j}\right),$$

kde b_{ij} je pravdepodobnosť výskytu aminokyselín i a j vo väzbe v β-listoch a f_i, f_j sú pravdepodobnosť výskytu jednotlivých aminokyselín v β-listoch. Táto reprezentácia popisuje s akou pravdepodobnosťou dané dve aminokyselinami interagujú v β-listoch v porovnaní s pravdepodobnosťou, s akou sa tieto dve aminokyseliny vyskytujú v β-listoch nezávisle na sebe.

Frekvencie výskytu aminokyselín pre vypočítanie potrebných pravdepodobností boli spočítané na trénovacích sadách pre pozitívne paralelné a antiparalelné príklady pre trojice väzieb (A5 PAR, A5 ANTIPAR). Z príkladov v týchto množinách sme spravili dve rôzne matice. V prvej, ktorú označíme *middle b matrix*, sme pre frekvencie výskytu spárovaných aminokyselín počítali len väzby medzi prostrednými aminokyselinami ($3 \sim 3$). V druhej s označením *all b matrix* sme počítali frekvencie výskytu aminokyselín vo všetkých troch väzbách v príklade. Výsledky testovania klasifikačnej sily pre tieto matice uvedieme v časti 4.4.3.

Nevýhodou takejto skórovacej matice je, že vyjadruje štatistický vzťah len medzi danými dvoma aminokyselinami, ale neberie do úvahy kontext sekvencie, v ktorej sa tieto dve aminokyseliny nachádzajú. Z tohto dôvodu tento spôsob určovania, či aminokyseliny v β-listoch interagujú alebo nie, nemusí poskytovať dostatočnú úspešnosť pri klasifikácii vzoriek pre potreby nášho deskriptora. Preto sme otestovali aj výpočtovo náročnejšiu metódu

(SVM), ktorá umožnuje zahrnúť pri do analýz aj susediace aminokyseliny v blízkosti interagujúceho páru.

4.3 Support vector machines (SVM)

4.3.1 Princíp SVM

Support vector machines (Vapnik, 1995) je metóda, ktorá slúži na klasifikáciu vstupov do dvoch rôznych tried (Friedman et al., 2001). Vstupy pre trénovanie tohto klasifikátora sú dvojice $(\vec{x_1}, y_1), (\vec{x_2}, y_2), ...(\vec{x_n}, y_n)$, kde $\vec{x_i} \in R^p$ sú vektory bodov v p-rozmernom priestore a $y_i \in \{-1, 1\}$ označuje príslušnosť k triede. Cieľom klasifikátora je nájsť vyjadrenie rozdelenia bodov do ich patričných tried, ktoré čo možno najlepšie zachytí rozdielne vlastnosti týchto tried a tak umožní klasifikáciu ďalších vstupov.

Túto úlohu riešia SVM nájdením nadroviny, ktorá oddeľuje vstupné body v priestore, určenej parametrami vo vektore $\vec{\beta}$.

$$\{\vec{x}\colon f(\vec{x}) = \langle\vec{\beta}, \vec{x}\rangle + \beta_0 = 0\}$$

Body sú klasifikované pomocou tejto nadroviny na základe toho, v ktorej časti priestoru rozdeleného nadrovinou ležia. Na jednej strane ($f(\vec{x}) > 0$) ležia body zaradené do prvej triedy a na opačnej strane ($f(\vec{x}) < 0$) budú body prislúchajúce druhej triede, takže pri správnej separácii bude platiť $y_i f(\vec{x_i}) > 0$ pre všetky vstupy. Z tohto pohľadu sa na funkciu $f(\vec{x})$ dá pozerať ako na skóre pre príslušnosť vstupu $\vec{x}$ do prvej triedy. Čím je hodnota $f(\vec{x})$ väčšia, tým je vstupný bod $\vec{x}$ vzdialenejší od deliacej nadroviny a tým je hlbšie v oblasti pre body patriace do prvej triedy.

Vstupné body je možné do jednotlivých tried rozdeliť množstvom rôznych nadrovín. Z týchto nadrovín chceme nájsť takú, ktorá bude správne klasifikovať aj ďalšie body z priestoru. Preto je snahou nájsť nadrovinu, ktorá bude maximalizovať vzdialenosť medzi vstupmi z opačných tried, ktoré sú najbližšie k deliacej rovine, a tak vytvoriť najširší pás bez bodov medzi oboma triedami. Body ležiace na hraniciach tohto pásu sa nazývajú *podporné vektory (support vectors)*. Problém nájdenia takejto roviny sa dá formulovať ako optimalizačná úloha:

$$\begin{aligned} &\underset{\vec{\beta},\beta_0}{minimalizuj} && \frac{1}{2}||\vec{\beta}|| \\ &za\ podmienok && y_i(\langle\vec{\beta},\vec{x_i}\rangle + \beta_0) \geq 1 \quad \forall i = 1..n \end{aligned}$$

kde podmienky zabezpečia správnu klasifikáciu všetkých bodov a minimalizácia $||\vec{\beta}||$ je ekvivalentná s maximalizáciou vzdialenosti medzi vstupmi najbližšie k nadrovine (Friedman et al., 2001). Táto formulácia predstavuje kvadratický optimalizačný problém, ktorý sa dá riešiť štandardnými numerickými metódami a nájsť tak optimálne separujúcu nadrovinu za predpokladu, že vstupné body sú lineárne separovateľné.

Tento spôsob klasifikácie má riešenie len pre lineárne separovateľné problémy, pretože nepripúšťa chybu pri klasifikovaní vstupných bodov. V reálnych situáciách sú vstupné údaje zriedka lineárne separovateľné a klasifikátor musí pripúšťať aj určitú mieru vstupov umiestnených na zlej strane nadroviny. Toto umožní použitie metódy na širšiu množinu problémov.

K pôvodnej formulácii optimalizačného problému sa pridajú chybové premenné ϵ, ktoré zrelaxujú podmienky pre korektné klasifikovanie vstupných bodov. Podmienky v probléme formulujeme ako $y_i(\langle\vec{\beta},\vec{x_i}\rangle + \beta_0) \geq 1 - \epsilon_i$, kde ϵ_i predstavuje chybu v klasifikácii i-teho vstupu. Celková chyba pri klasifikácii je vyjadrená sumou $\sum_{i=1}^{n} \epsilon$, ktorá je s určitou váhou C pridaná do minimalizovaného výrazu. Takto upravený problém má tvar:

$$\begin{aligned} &\underset{\vec{\beta},\beta_0}{minimalizuj} && \frac{1}{2}||\vec{\beta}|| + C\sum_{i=1}^{n}\epsilon_i \\ &za\ podmienok && y_i(\langle\vec{\beta},\vec{x_i}\rangle + \beta_0) \geq 1 - \epsilon_i \quad \forall i = 1..n. \end{aligned}$$

Premenná C vyjadruje váhu, ktorú prikladáme zle klasifikovaným vstupom a vplýva na veľkosť a položenie oddeľujúceho pásu, ktorý podporné vektory vyznačujú okolo nadroviny. Malá hodnota váhy C umožňuje, aby viac vstupov bolo položených na zlej strane oddeľujúcej nadroviny a rozširuje tak tento pás. Veľká hodnota C výraznejšie penalizuje chybnú klasifikáciu a tak upredňostňuje riešenia, ktoré vyznačujú užší pás.

Formulácia pomocou Lagrangeových multiplikátorov umožňuje vyjadriť váhy $\vec{\beta}$ ako sumu $\vec{\beta} = \sum_{i=1}^{n} y_i\alpha_i\vec{x_i}$ a deliaca nadrovina má v tomto vyjadrení tvar $f(x) = \sum_{i=1}^{n} y_i\alpha_i\langle\vec{x_i}\vec{x}\rangle$.

Pomocou tejto formulácie sa dá modifikovaný problém vyjadriť v takzvanom *duálnom* tvare (Friedman et al., 2001) :

$$\begin{aligned} &\underset{\vec{\alpha}}{\textit{maximalizuj}} && \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &\textit{za podmienok} && \sum_{i=1}^{n} y_i \alpha_i = 0, \forall i : 0 \leq \alpha_i \leq C \end{aligned}$$

Pre *podporné vektory* x_i v tomto vyjadrení platí $\alpha_i > 0$, ostatné body majú $\alpha_i = 0$, a teda nevplývajú na určenie polohy deliacej nadroviny. Problém nájdenia nadroviny takto rozšírený o možnosť nesprávnej klasifikácie časti vstupov je možné použiť aj na lineárne neseparovateľné problémy, pričom ale samotná deliaca hranica je stále lineárna, čo limituje vyjadrovaciu silu tohto prístupu.

Avšak formulácia úlohy v duálnom tvare umožňuje zapojiť ďalšiu metódu - transformáciu $\phi(x)$, ktorá premietne vstupy do priestoru s vyšším rozmerom. Po transformácii je teda potrebné spočítať skalárny súčin $\langle \phi(x_i), \phi(x_j) \rangle$ premenných premietnutých do priestoru s väčšou dimenziou a v tomto priestore určiť lineárnu deliacu nadrovinu. Ak je transformácia ϕ nelineárna, po prevedení nadroviny naspäť do pôvodného priestoru dostaneme nelineárnu hranicu medzi triedami. Pri dobre zvolenej transformácii môže tento prístup výrazne zväčšiť klasifikačnú silu aj keď transformácia bodov do nového priestoru a naspäť môže byť výpočtovo náročná.

Vhodnou voľbou transformácie je možné obísť problém výpočtovej náročnosti. V duálnom tvare problému vystupujú vstupné body x_i len v skalárnom súčine $\langle x_i, x_j \rangle$ s ostatnými vstupnými bodmi x_j a pre nájdenie nadroviny tak stačí poznať výsledok tohto skalárneho súčinu. Definujeme *kernelovú* funkciu ako $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. Ak vieme efektívne spočítať túto funkciu priamo, vyhneme sa výpočtovo náročnej transformácii bodov x_i a x_j. Túto kernelovú funkciu je možné dosadiť priamo do duálneho tvaru problému a tak dostaneme klasifikátor, ktorý je schopný vytvárať nelineárne deliace hranice v priestore vstupných bodov. Po nahradení skalárneho súčinu kernelovou funkciou je deliaca nadrov-

ina vyjadrená ako

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(\vec{x}, \vec{x_i}) + \beta_0$$

a rovnakým spôsobom sa zmení aj formulácia optimalizačného problému. Výpočtová zložitosť klasifikácie pomocou SVM s kernelovou funkciou, ako aj jej kvalita, závisí od výberu kernelovej funkcie. Štandardné kernelové funkcie sú napríklad:

- **Polynomiálny kernel** - $K(\vec{x_i}, \vec{x_j}) = (\langle \vec{x_i}, \vec{x_j} \rangle + k)^d$. Polynomiálny kernel s parametrami k a d. Čím väčší je parameter d, tým väčšiu dimenziu má priestor, do ktorého sa transformujú pôvodné vstupy.

- **Radiálna bázová funkcia** - $K(\vec{x_i}, \vec{x_j}) = e^{(-\frac{1}{\sigma}||\vec{x_i} - \vec{x_j}||^2)}$. Funkcia, ktorá vychádza z Gaussovho rozdelenia. Parameter σ určuje šírku Gaussovej krivky použitej na namapovanie vstupov do vyšších rozmerov.

Pri praktickom použití SVM je nutné okrem vhodnej kernelovej funkcie aj určiť jej parametre, pretože tie môžu výrazne ovplyvniť časovú zložitosť výpočtu alebo viesť k preučeniu klasifikátora. SVM boli v praxi úspešne použité na riešenie biologických problémov, napríklad aj na klasifikáciu funkcie proteínov (Leslie et al., 2002).

4.3.2 SVM pre rozpoznávanie spárovaných aminokyselín

Schopnosť SVM klasifikovať lineárne neseparovateľné množiny vstupov nelineárnou deliacou hranicou sme využili pre natrénovanie špecifických SVM pre rozoznávanie spárovaných aminokyselín. Naším cieľom je vytvoriť klasikátor, ktorý bude ohodnocovať dve aminokyseliny z hľadiska očakávania vytvorenia hydrogénovej väzby medzi nimi. Keďže spárované aminokyseliny sa v interagujúcich β-listoch vyskytujú v rade za sebou, kontext okolitých aminokyselín môže pomôcť lepšie identifikovať aminokyseliny viazané hydrogénovými väzbami. Na konštrukciu takéhoto klasifikátora použijeme trénovacie sady s pozitívnymi aj negatívnymi príkladmi, ktoré predstavujú úseky so spárovanými a nespárovanými aminokyselinami. Pre tieto príklady je potrebné určiť *atribúty*, ktorými sú charakterizované v SVM.

model	trénovacia sada	P	N
M5 PAR	A5 PAR	70000	350000
M5 ANTIPAR	A5 ANTIPAR	70000	350000
M5 MIX	A5 PAR + A5 ANTIPAR	140000	700000
M3 PAR	A3 PAR	70000	350000
M3 ANTIPAR	A3 ANTIPAR	70000	350000
M5 B PAR MIX	A5 PAR + A5 B PAR	140000	700000
M5 B ANTIPAR MIX	A5 PAR + A5 B ANTIPAR	140000	700000

Tabuľka 4.3: Prehľad natrénovaných modelov. Stĺpec P označuje počet pozitívnych príkladov v použitej trénovacej sade. Stĺpec N počet negatívnych príkladov v tejto sade.

Aby tieto SVM klasifikátory mali čo najlepšiu rozlišovaciu silu klasifikátora, musíme testovaním nájsť najvhodnejšie hodnoty parametrov pre použité kernelové funkcie v zostavených SVM. Počet použitých čŕt a veľkosť parametrov výrazne ovplyvňuje okrem kvality klasifikácie aj časovú náročnosť trénovania SVM a klasifikácie ďalších vstupov. Keďže budeme pri hľadaní optimálneho zarovnania deskriptoru a sekvencie ohodnocovať veľké množstvo párov aminokyselín, aby bol natrénovaný SVM využiteľný v našom deskriptore, musí poskytovať dostatočne dobrú separáciu v prijateľnom čase.

Pre náš deskriptor sme na základe už popísaných trénovacích sád (A5 PAR, A5 ANTIPAR, A5 B PAR, A5 B ANTIPAR) s interagujúcimi aminokyselinami v β-listoch vytvorili niekoľko SVM klasifikátorov s rôznym počtom atribútov (implementácia SVM Light (Joachims, 1999)). Podľa trénovacích sád a použitých atribútov môžeme rozdeliť vytvorené SVM klasifikátory na tri skupiny (viď tabuľka 4.3):

- **SVM pre spárované trojice aminokyselín** - Na natrénovanie prvej skupiny SVM sme použili trénovacie sady A5 PAR a A5 ANTIPAR. Príklady v týchto množinách sú páry úsekov sekvencie s dĺžkou päť aminokysein. V pozitívnych príkladoch obsahovali tri hydrogénové väzby medzi aminokyselinami v týchto dvoch úsekoch. V prípade paralelných príkladov to sú páry medzi aminokyselinami (1∼1, 3∼3, 5∼5), pre antiparalelné príklady (1∼5, 3∼3, 5∼1). Negatívny príklad sú dve pätice aminokyselín, medzi ktorými nie je žiadna väzba.

  Tieto vstupy sme reprezentovali sadou atribútov. Prvý atribút je skóre z vyššie

popísanej skórovacej matice pre prostredný pár aminokyselín (3∼3). Ďalšie atribúty binárne kódujú jednotlivé aminokyseliny vo vstupnom páre úsekov. Každá aminokyselina je reprezentovaná dvadsiatimi atribútmi, z ktorých má práve jedna hodnotu 1, ostatné sú 0. Takýmto spôsobom je vyjadrených všetkých 5 aminokyselín z oboch úsekov, čo znamená, že jeden vstup má 201 atribútov $(1 + 2 \times 5 \times 20)$.

Z takto reprezentovaných trénovacích množín sme vytvorili 3 SVM modely. Prvý model, ktorý označíme *M5 PAR*, bol natrénovaný na množine A5 PAR (70000 pozitívnych a 5×70000 negatívnych príkladov) na rozlišovanie pozitívnych paralelných príkladov od neinteragujúcich príkladov. Druhý model M5 ANTIPAR (70000 pozitívny a 5×70000 negatívnych príkladov) rozlišuje pozitívne antiparalelné príklady od neinteragujúcich. Tretí model M5 MIX mal ako trénovaciu množinu zjednotenie množín A5 PAR a A5 ANTIPAR, pričom paralelné aj antiparalelné pozitívne príklady boli použité ako pozitívne aj v tomto modely. Takže tento model bol natrénovaný na 140000 pozitívnych a 5×140000 negatívnych príkladov. Tento model rozlišuje interagujúci príklad v ľubovoľnom smere od neinteragujúceho príkladu.

Každý z týchto modelov bol natrénovaný s polynomiálnym a radiálnym kernelom. Pre polynomiálny kernel boli použité parametre $d = 3, 5, 7, 9$ a radiálny kernel bol natrénovaný s parametrom $\sigma = 0.1, 0.3, 0.5, 07$. To znamená, že v tejto skupine máme 24 (3×8) modelov.

- **SVM pre spárované trojice aminokyselín s vynechanými pozíciami** - Pri hľadaní najlepšieho zarovnávania deskriptora ku sekvencii je nutné ohodnotiť pomocou SVM veľké množstvo párov úsekov, čo pre 201 atribútov s polynomiálnym kernelom vyššieho rádu znamená veľkú časovú náročnosť. Ak by sme mali predpočítané skóre pre všetky možné vstupy, zrýchlilo by to hľadanie zarovnania. Vo vstupoch sú reprezentované páry pätíc aminokyselín, čo znamená, že možných vstupov je 20^{10}. Toto je príliš veľa na predpočítanie, preto sme skúsili z príkladov v trénovacích množinách A5 PAR a A5 ANTIPAR vypustiť nespárované aminokyseliny, teda aminokyseliny na pozíciách 2 a 4. Vstupy teraz kódujú iba páry trojíc

aminokyselín, čo znamená 20^6 možných vstupov. Na vstupoch upravených takýmto spôsobom sme natrénovali SVM pre rozoznávanie paralelných a antiparalelných príkladov, ktoré označíme M3 PAR a M3 ANTIPAR. Aj v tomto prípade sme použili dva kernely (polynomiálny a radiálny), každý so štyrmi hodnotami kernelového parametru ($d = 3, 5, 7, 9$ a $\sigma = 0.1, 0.3, 0.5, 07$). V tejto skupine je teda 16 (2×8) modelov.

- **SVM pre spárované aminokyseliny so spárovaným susedom** - SVM pre spárované trojice neberie do úvahy aminokyseliny na okrajoch spárovaných β-listov, ktoré majú v okolí len jednu ďalšiu spárovanú aminokyselinu. Aminokyseliny na okrajových pozíciách neboli medzi pozitívnymi vstupmi a preto môžu byť zle klasifikované týmito SVM, čo môže negatívne ovplyvniť celkový prínos do skóre zarovnania deskriptoru ku sekvencii. Preto sme vytvorili ďalšie SVM, ktoré zachytávajú spárované aminokyseliny aj na krajoch, aj v strede interagujúcich β-listov.

  Vytvoríme samostatné SVM pre oba smery (paralelný a antiparalelný) spárovania β-listov, ktoré obsahujú medzi pozitívnymi príkladmi aj okrajové príklady. To znamená páry sekvencií, v ktorých sú len dve väzby. Jedna medzi strednými aminokyselinami (3∼3) a jedna na okraji. Takéto príklady sú v trénovacích množinách A5 B PAR a A5 B ANTIPAR. Pre každý smer vytvoríme modely, ktoré budú natrénované na zjednotení trénovacej množiny s trojicami väzieb a trénovacej množiny pre okrajové prípady. Takže rozšírený paralelný model M5 B PAR MIX bude natrénovaný na množine A5 PAR MIX (zjednotenie množín A5 PAR a A5 B PAR). Analogicky antiparalelný model M5 B ANTIPAR MIX je natrénovaný na zjednotení množín A5 ANTIPAR a A5 B ANTIPAR. Takže tieto modely boli natrénované na sade obsahujúcej 140000 pozitívnych a 5×140000 negatívnych príkladov.

  Opäť boli modely natrénované s dvoma kernelmi a štyrmi hodnotami pre ich parametre ($d = 3, 5, 7, 9$ a $\sigma = 0.1, 0.3, 0.5, 07$), to znamená, že v tejto skupine je tiež 16 modelov.

Aby sme mohli porovnať, ktoré modely a s akými parametrami poskytujú najlepšiu klasifikačnú silu pre odlíšenie úsekov s interagujúcimi a neinteragujúcimi aminokyselinami,

urobili sme sériu testov na testovacích množinách, ktorých výsledky popíšeme v nasledujúcej časti.

4.4 Testovanie klasifikátorov

4.4.1 ROC krivky

Nech testovacia sada pozostáva z p pozitívnych a n negatívnych príkladov. Nech klasifikačná metóda, ktorú chceme ohodnotiť, z týchto p pozitívnych príkladov označí správne a príkladov za pozitívne (true positives) a b príkladov mylne za negatívne (false negatives). Podobne nech z n negatívnych príkladov správne klasifikuje c príkladov ako negatívne príklady (true negatives) a mylne d príkladov ako pozitívne (false positives). Toto rozdelenie je zobrazené v tabuľke 4.4.

Senzitivitou nazveme pomer medzi správne klasifikovanými pozitívnymi príkladmi a (true positives) a všetkými pozitívnymi príkladmi p ($SN = a/p$). *Specificitou* nazveme pomer medzi správne klasifikovanými negatívnymi príkladmi d (true negatives) a všetkými negatívnymi príkladmi p ($SP = d/n$).

Úspešnosť jednotlivých klasifikátorov sme vyhodnocovali pomocou ROC (Receiver operating characteristic) kriviek. ROC krivka vyjadruje vzťah medzi senzitivitou a specificitou pre daný klasifikátor. Na y-novej osi je senzitivita, to znamená číslo z intervalu $\langle 0, 1\rangle$, ktoré vyjadruje akú časť z pozitívnych príkladov správne identifikoval daný klasifikátor. Na x-ovej je 1-specificita, čiže tzv. false positive rate, to znamená číslo z intervalu $\langle 0, 1\rangle$, ktoré vyjadruje akú časť z negatívnych príkladov klasifikátor mylne označil za pozitívne príklady. ROC krivka teda vyjadruje, akú časť pozitívnych príkladov klasifikátor odhalí pri danej miere mylne označených pozitívnych príkladov. Pri analýzach hľadáme model, ktorý dosiahne veľkú senzitivitu pri malom počte falošne pozitívnych príkladov.

4.4.2 Výsledky jednotlivých modelov

Na siedmych rôznych trénovacích sadách boli natrénované SVM modely s dvoma rôznymi kernelmi, každý so štyrmi parametrami, to znamená spolu 56 ($7 \times 2 \times 4$) rôznych modelov.

	klasifikované ako pozitívne	klasifikované ako negatívne
Pozitívne príklady $p = (a + b)$	a (true positives)	b (false negatives)
Negatívne príklady $n = (d + c)$	c (false positives)	d (true negatives)

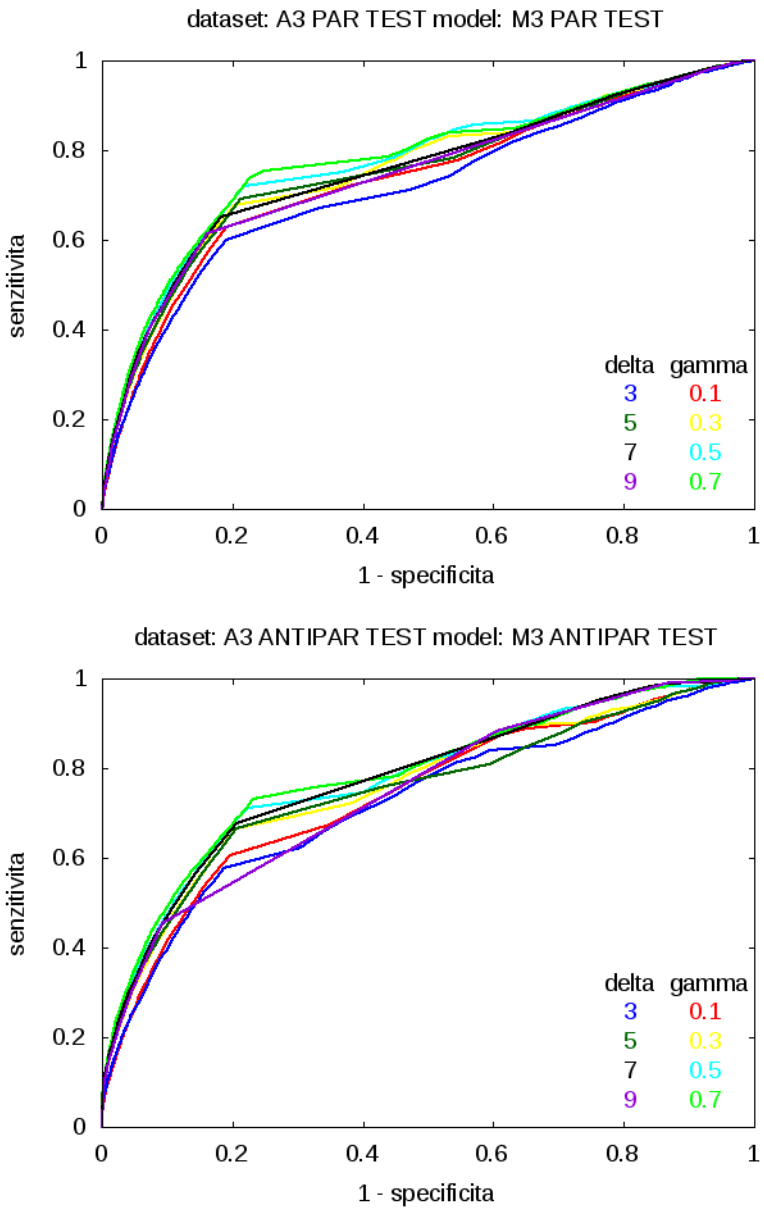
Tabuľka 4.4: Rozdelenie testovacej množiny podľa výsledkov predikcie daným klasikátorom.

Prvým testom bolo zistiť, s akými parametrami dosahujú modely najlepšiu úspešnosť pri klasifikácii prislúchajúcich testovacích množín. Testovacie sady A3 ANTIPAR TEST, A3 PAR TEST, A5 ANTIPAR TEST, A5 PAR TEST, A5 ANTIPAR B TEST, A5 PAR B TEST a A5 MIX TEST sme ohodnotili prislúchajúcimi modelmi, štyrmi s radiálnym a štyrmi s polynomiálnym kernelom. Výsledky tejto klasifikácie sme zobrazili pomocou ROC kriviek, ktoré vyjadrujú klasifikačnú silu modelov s jednotlivými parametrami.

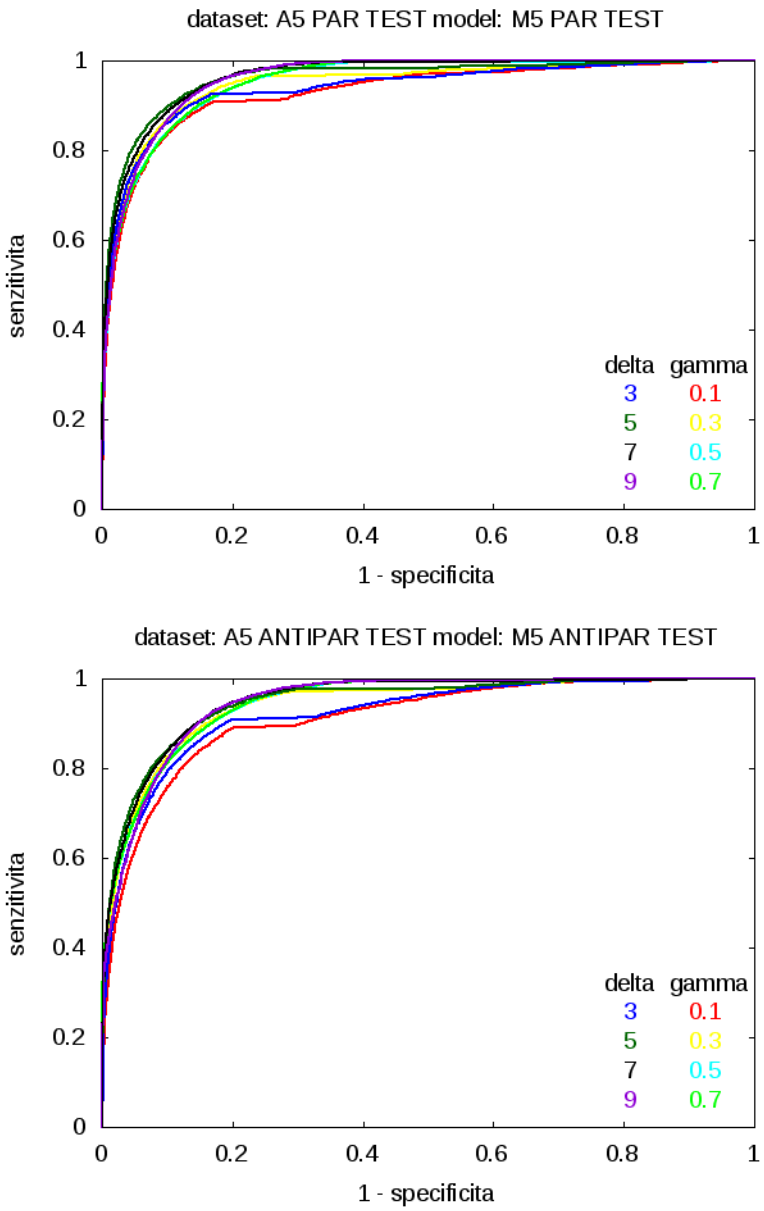
Na obrázku 4.3 sú zobrazené výsledky pre modely natrénované na trénovacích sadách A3 PAR a A3 ANTIPAR (paralelné a antiparalelné spárované trojice aminokyselín s vynechanými neinteragujúcimi pozíciami). Pre paralelné príklady dosiahol najlepšie výsledky model s radiálnym kernelom s parametrom gamma = 0.7. Naopak, najhorší výsledok v tomto prípade mal model s polynomiálnym kernelom s parametrom delta = 9. Pre príklady s antiparalelnými väzbami je opäť najúspešnejší model s radiálnym kernelom a parametrom gamma = 0.7, ale najhoršie výsledky na tejto testovacej sade dosiahol model s polynomiálnym kernelom a parametrom delta = 3.

Ďalšie testované modely boli natrénované na trénovacích sadách A5 PAR a A5 ANTIPAR (paralelné a antiparalelné spárované trojice aminokyselín). Výsledky pre tieto modely sú na obrázku 4.4. V tomto prípade sú výsledky modelov veľmi vyrovnané, pričom model s polynomiálnym kernelom a parametrom delta = 5 má najvyššiu senzitivitu pri nízkom podiele falošne pozitívnych príkladov pre paralelné aj antiparalelné príklady. Najhoršie výsledky v oboch prípadoch mal model s radiálnym kernelom s parametrom gamma = 0.1.

Trénovacie sady A5 PAR B a A5 ANTIPAR B sú sady A5 PAR a A5 ANTIPAR rozšírené o okrajové prípady, v ktorých chýbala jedna väzba medzi aminokyselinami na



Obr. 4.3: ROC krivky pre modely natrénované na trénovacích sadách A3 PAR a A3 ANTIPAR.



Obr. 4.4: ROC krivky pre modely natrénované na trénovacích sadách A5 PAR a A5 ANTIPAR.

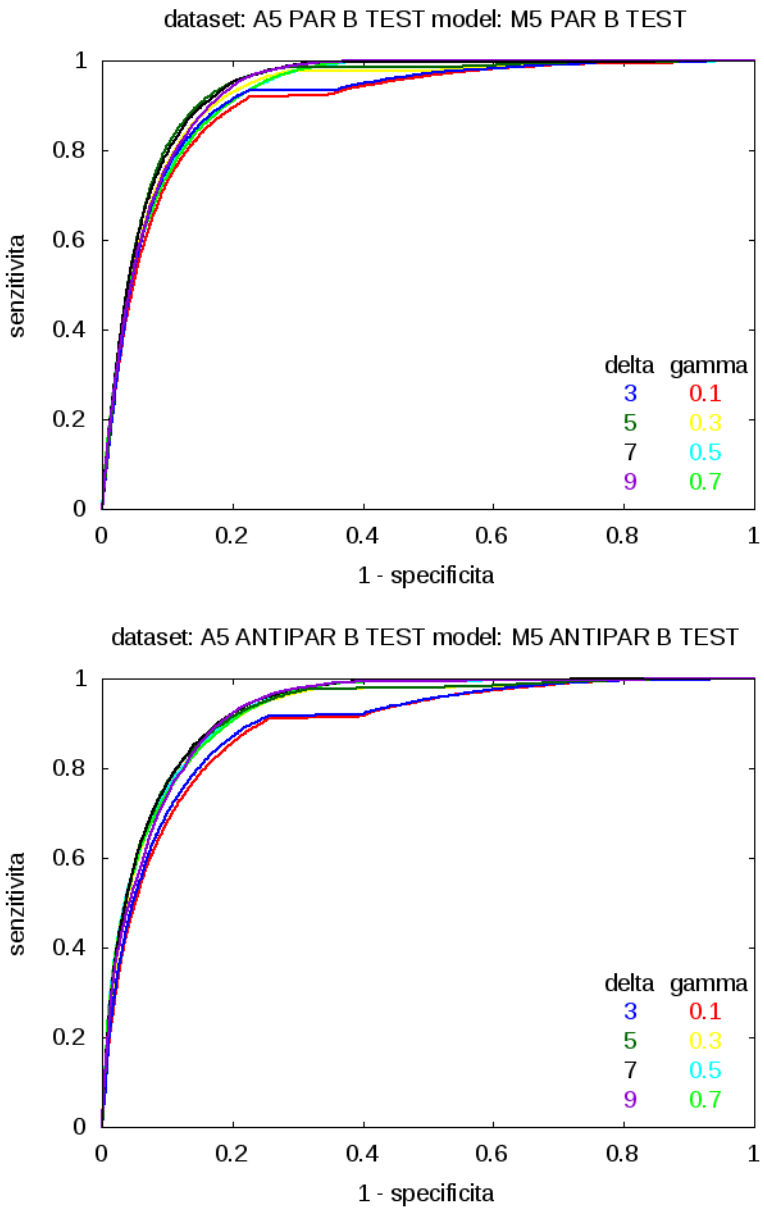
začiatku alebo konci vzorky. Interagujú teda len dva páry aminokyselín v príklade (pár $3 \sim 3$ a ďalší interagujúci pár). Výsledky modelov natrénovaných na týchto sadách sú na obrázku 4.5. V tomto prípade opäť najhoršie výsledky pre oba smery dosahujú modely s parametrami delta = 3 (polynomiálny kernel) a gamma = 0.1 (radiálny kernel). Výsledky ostatných modelov sú vyrovnané, pričom najlepší výsledok poskytujú modely s polynomiálnym kernelom pre parameter delta = 5 a 7.

Posledná skupina modelov, M5 MIX, bola natrénovaná na zjednotení množin A5 PAR a A5 ANTIPAR. Tento model teda bol natrénovaný na príkladoch trojíc spárovaných aminokyselín, ktoré nerozlišovali smer. Tieto modely boli otestované na zjednotení testovacích množín A5 PAR TEST a A5 ANTIPAR TEST a výsledky týchto testov sú zobrazené na obrázku 4.6. V tomto prípade modely s parametrami delta = 3 a gamma = 0.1 majú v porovnaní s ostatnými modelmi vyššiu senzitivitu pri miere falošne pozitívnych príkladov pod 0.1. Ostatné modely však dosahujú vyššiu senzitivitu pri miere falošne pozitívnych príkladov väčšej ako 0.2. Z týchto modelov najrýchlejšie dosiahne najvyššiu senzitivitu model s polynomiálnym kernelom a parametrom delta = 7.

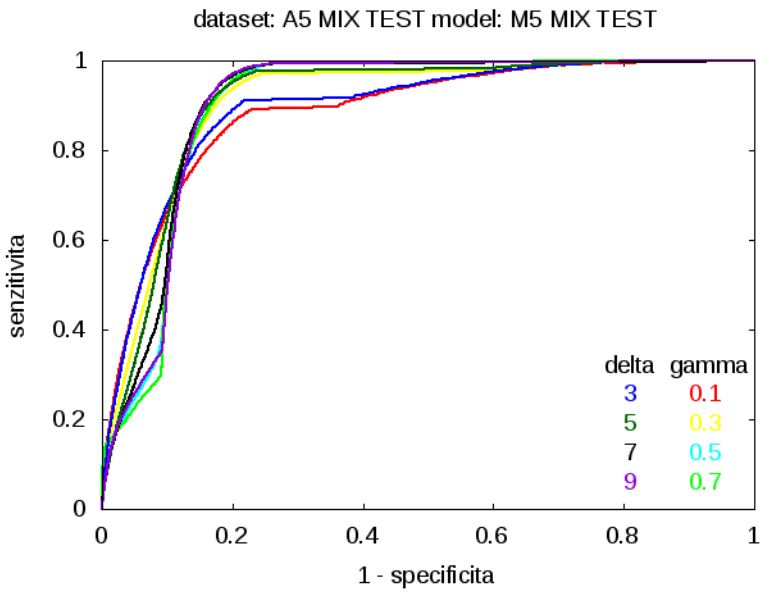
4.4.3 Výsledky porovnania modelov

Aby sme mohli tieto modely navzájom porovnať, musíme ich otestovať na spoločnej testovacej sade. Preto sme vytvorili testovaciu sadu A5 UNION TEST (zjednotenie sád A5 PAR B TEST a A5 ANTIPAR B TEST). Táto obsahuje 40000 pozitívnych príkladov, z čoho je 20000 príkladov paralelných väzieb a 20000 príkladov antiparalelných väzieb a 5×40000 negatívnych príkladov. Paralelné aj antiparalelné pozitívne príklady sa skladajú z 10000 príkladov pre spárované trojice a 10000 okrajových príkladov s bez jedného interagujúceho páru.

Príklady v sade A5 UNION TEST sme ohodnotili každým z natrénovaných modelov. Keďže sada obsahuje paralelné aj antiparalelné príklady, pre ohodnotenie vstupu skombinujeme paralelné a antiparalelné modely natrénované s rovnakými parametrami. Skóre vstupu je v tomto prípade maximum zo skór paralelným a antiparalelným modelom. Týmto spôsobom sme dostali klasifikátory, ktoré označíme podľa typu trénovacích sád, na ktorých



Obr. 4.5: ROC krivky pre modely natrénované na trénovacích sadách A5 PAR B a A5 ANTIPAR B (obsahujú aj okrajové aminokyseliny).

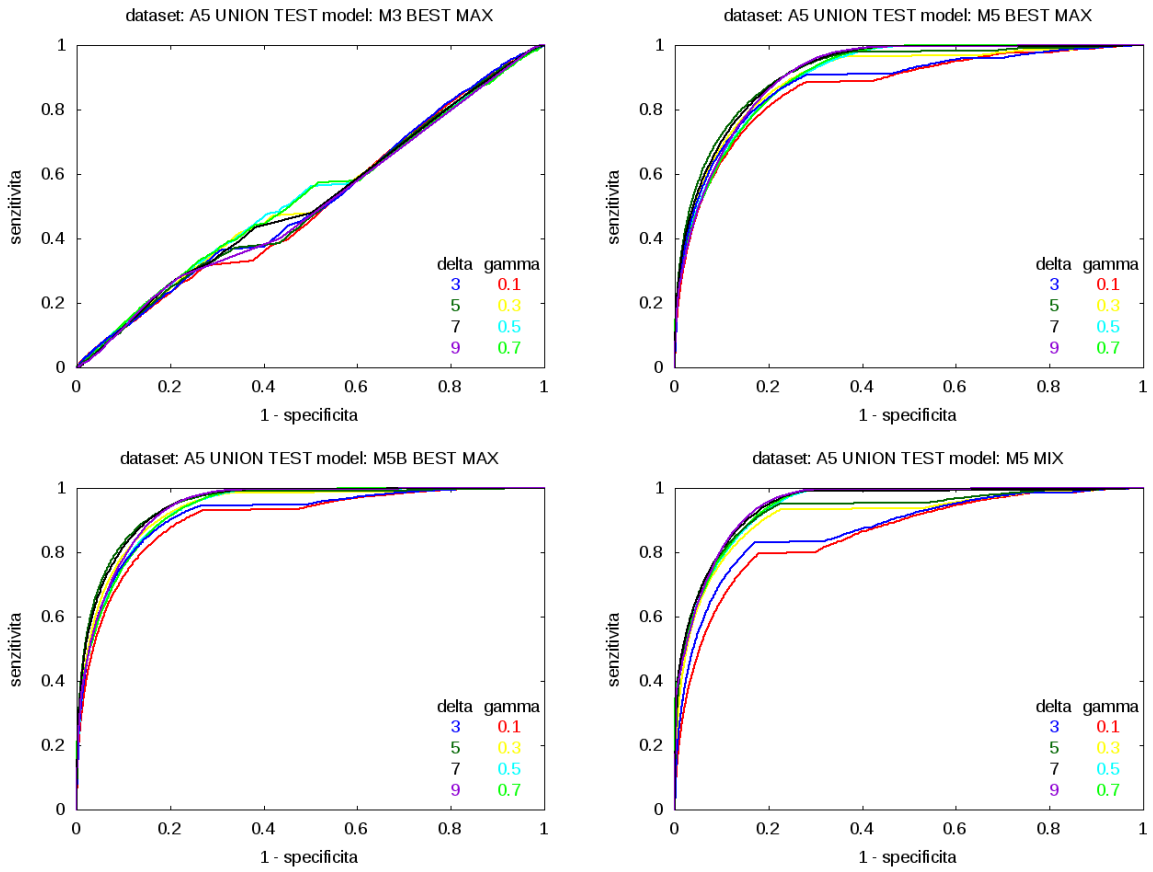


Obr. 4.6: ROC krivky pre model natrénovaný na trénovacej sade A5 MIX (zjednotenie sady A5 PAR a A5 ANTIPAR).

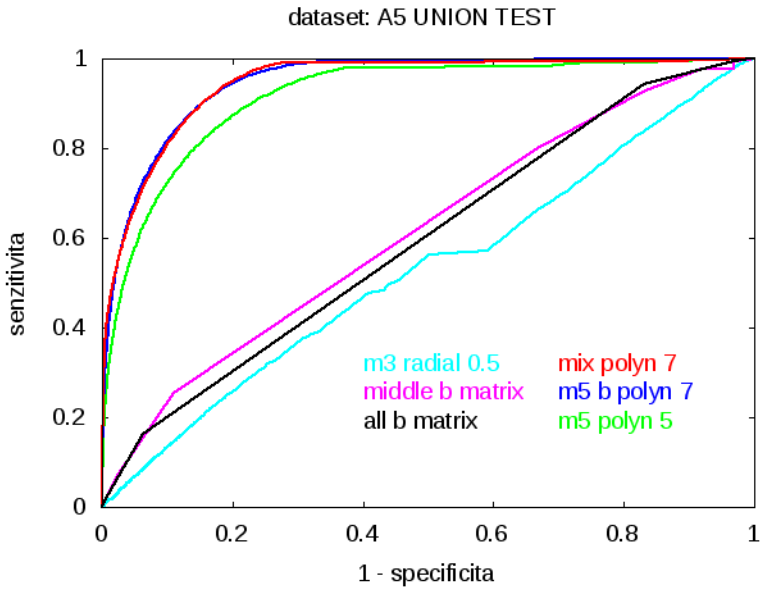
boli natrénované SVM modely (M3 BEST MAX, M5 BEST MAX, M3 BEST MAX). Na obrázku 4.7 sú zobrazené výsledky jednotlivých klasifikátorov na testovacej sade A5 UNION TEST.

Najhorší výsledok dosiahli klasifikátory zložené z paralelných a antiparalelných modelov M3, pri ktorých senzitivita stúpa rovnomerne s mierou falošne pozitívnych príkladov. To znamená, že tieto klasifikátory nedokážu odlíšiť pozitívne príklady od negatívnych. V prípade klasifikátorov zložených z M5 a M5 B modelov sú výsledky výrazne lepšie, pričom v oboch prípadoch najlepšie výsledky dosiahli modely s polynomiálnym kernelom a parametrami delta = 5 a 7. Najhoršie výsledky majú opäť modely s radiálnym kernelom a parametrom gamma = 0.1.

Najlepšie klasifikátory z jednotlivých skupín sme ďalej porovnali s ROC krivkami pre skórovacie matice popísané v časti 4.2. Výsledky tohto porovnania sú zobrazené na obrázku 4.8. Z porovnania vyplýva, že obe skórovacie matice (middle b matrix, all b matrix), rovnako ako klasifikátor zložený z M3 modelov, nemajú dostatočnú klasifikačnú silu na odlíšenie pozitívnych príkladov od negatívnych. Výsledky klasifikátora zloženého z mode-



Obr. 4.7: ROC krivky pre modely testované na množine A5 UNION TEST.



Obr. 4.8: ROC krivky pre model natrénovaný na trénovacej sade A5 MIX (zjednotenie sady A5 PAR a A5 ANTIPAR).

lov M5 sú výrazne lepšie. Klasifikátory s modelmi M5 B, ktoré boli natrénované na sadách rozšírených o okrajové prípady, majú ešte mierne lepšie výsledky. Podobné výsledky dosiahol aj model M5 MIX.

4.5 Záver

Na základe porovnania prezentovaného v časti 4.4.3 sme sa rozhodli použiť pre skórovanie očakávaných hydrogénových väzieb medzi aminokyselinami SVM modely M5 B PAR a M5 B ANTIPAR s polynomiálnym kernelom s parametrom delta = 7.

SVM pre skórovanie hydrogénových väzieb medzi aminokyselinami je posledným chýbajúcim stavebným prvkom skórovacej schémy popísanej v kapitole 3, ktorou môžeme ohodnotiť zarovnanie deskriptoru ku sekvencii. Pre identifikáciu vzťahu homológie medzi proteínmi potrebujeme metódu ako nájsť zarovnanie s najlepším skóre. V nasledujúcich dvoch kapitolách uvedieme dve metódy, ktoré riešia problém zarovnania s najlepším skóre.

Kapitola 5

Zarovnávanie deskriptorov pomocou celočíselného lineárneho programovania

V kapitole 3 sme definovali problém identifikácie ortológov pomocou hľadania s deskriptormi, ktoré boli zostavené odborníkmi, tak aby vystihovali dôležité štrukturálne vlastnosti. Našou úlohou je nájsť poďla danej skórovacej schémy najlepšie zarovnanie daného deskriptoru k vstupnej sekvencii. Jedná sa teda o optimalizačnú úlohu, v ktorej maximalizujeme skóre zarovnania. Viacero podobných úloh je NP-ťažkých, napríklad protein-threading (Lathrop, 1994) alebo vyhľadávanie RNA motívov (Rampasek, 2011). Keďže naša úloha je štruktúrovaná veľmi podobne, je predpoklad, že aj naša úloha je NP-ťažká.

V tejto kapitole ukážeme, ako riešiť našu úlohu jednou zo štandardných metód na riešenie ťažkých úloh: celočíselným lineárnym programovaním. Ukážeme, že kým pre pozitívne príklady je tento spôsob použiteľný na reálnych dátach, pri negatívnych príkladoch naráža na vysokú výpočtovú náročnosť.

5.1 Celočíselné lineárne programovanie

Lineárne programovanie je matematická metóda, ktorá sa používa na riešenie optimalizačných problémov. Pri riešení touto metódou sa problém vyjadrí ako funkcia, ktorá musí spĺňať sadu daných podmienok a optimálne riešenie predstavuje globálny extrém (minimum alebo maximum) tejto funkcie. Na riešenie takto sformulovaného problému sa používajú štandardné metódy pre výpočet lineárneho programu. Napríklad metóda Simplex (Dantzig,

1951), ktorá vypočíta najlepšie prípustné riešenie tak, že začne zo základného prípustného riešenia a prehľadáva susedov tohto riešenia v priestore prípustných riešení. Konečným počtom iterácií sa takýmto spôsobom nájde optimálne riešenie.

Lineárny program je teda formuláciou optimalizačného problému v tvare:

$$\begin{aligned} \underset{\vec{x}}{maximalizuj} \quad & F(\vec{x}) = \sum_{i=1}^{n} e_i x_i \\ za\ podmienok \quad & g_j(\vec{x}) \geq b_j, \;\; g_j(\vec{x}) = \sum_{i=1}^{n} g_{ij} x_i \\ & \forall i, j : x_i, e_i, g_{ij}, b_j \in \Re \end{aligned}$$

$F(\vec{x})$ predstavuje tzv. *účelovú funkciu*, lineárnu funkciu premenných $\vec{x}$, ktorej extrém sa snažíme nájsť. Faktory e_i určujú akou mierou prispievajú jednotlivé premenné do účelovej funkcie F. Sada M podmienok $G = \{g_j(\vec{x}) \geq b_j \;, j = 1..M\}$ špecifikuje pomocou lineárnych rovníc a nerovníc prípustné riešenia. Podmienky sa dajú vyjadriť aj vo forme súčinu matice s parametrami podmienok g_{ij} a vektora premenných $\vec{x}$. Ak premenné $\vec{x}$ môžu byť len z množiny celých čisel, hovoríme o *celočíselnom (Integer)* lineárnom programe (ILP). Riešenie tohto typu lineárnych programov je NP-ťažký problém (Schrijver, 1998). Pre problémy spojené s rozhodovaním sú vhodné *binárne* celočíselné programy, v ktorých môžu premenné nadobúdať len hodnoty 0 alebo 1. Práve tento typ lineárneho programovania sme použili na určovanie najlepšieho zarovnania deskriptora ku sekvencii.

Na riešenie celočíselných programov sú vyvinuté viaceré metódy, ktoré spočívajú v prehľadávaní priestoru prípustných riešení. Metóda *branch-and-bound* (Wolsey, 1980) pri hľadaní optimálneho celočíselného riešenia vychádza z relaxovaného problému, bez obmedzenia na celočíselnosť premenných $\vec{x}$. Najprv sa tento problém vyrieši v tomto tvare (napríklad Simplex metódou), čo určí optimálne riešenie $\vec{x_0}$. V prípade, že všetky zložky tohto vektora sú celočíselné, máme optimálne riešenie aj pre celočíselný problém. V opačnom prípade musí existovať aspoň jedna zložka vektora $\vec{x_0}$, ktorá nie je celé číslo. Vyberieme jednu z takýchto zložiek a označíme ju x_{0_i}. Pre optimálne celočíselné riešenie $\vec{x}^*$ bude určite platiť buď $x_i^* \leq \lfloor x_{0_i} \rfloor$ alebo $x_i^* \geq \lceil x_{0_i} \rceil$. Pridaním jednej alebo druhej nerovnosti do pôvodného problému dostaneme dva nezávislé podproblémy, ktoré môžeme rekurzívne riešiť. Opäť

nájsť najprv neceločíselné riešenie a následne ďalšie rekurzívne vetvenie na podproblémy. Tento krok algoritmu sa nazýva *vetvenie (branching)*. Taktiež optimálne celočíselné riešenie $\vec{x}^*$ bude určite menšie ako $\vec{x_0}$ a určite väčšie alebo rovné najlepšiemu doposiaľ nájdenému celočíselnému riešeniu. Toto ohraničenie *(bound)* predstavuje horný a dolný odhad na optimálne celočíselné riešenie. Túto skutočnosť je zároveň možné využiť pre optimalizáciu výpočtu. Pri rekurzívnom prehľadávaní stromu riešení si pamätáme najlepšie doposiaľ nájdené celočíselné riešenie a neprehľadávame vetvy, ktoré budú obsahovať určite horšie riešenia. Prehľadávanie sa ukončí, keď sú vyhodnotené všetky podproblémy, výsledok je najlepšie celočíselné riešenie v prehľadávacom strome.

Rýchlosť dosiahnutia optimálneho riešenia do veľkej miery závisí od poradia, ako sú volené zložky vektora na vetvenie a akým sú vyhodnocované problémy v rekurzii. Implementácia celočíselného programu pomocou *riedkych (sparse) matíc* a množstvo heuristík použitých v rôznych solveroch (napríklad solver CPLEX (Cplex, 2010)) optimalizujú výpočet riešenia a umožňujú praktické použitie celočíselného programu pre optimalizačné problémy aj s veľmi veľkou množinou premenných.

Pre využitie celočíselného programu na hľadanie najlepšieho zarovnania deskriptoru a sekvencie bolo potrebné nájsť spôsob ako tento problém formulovať ako binárny lineárny program.

5.2 Problém zarovnania deskriptora ako celočíselný lineárny program

Naším cieľom je nájsť binárny lineárny program, ktorého riešenie určí najlepšie zarovnanie daného deskriptora k vstupnej sekvencii t.j. zarovnanie s najvyšším skóre podľa skórovacej schémy, ktorú sme popísali v kapitole 3. Úlohou je vlastne určiť, ktoré aminokyseliny sekvencie zodpovedajú štrukturálnym segmentom a sekvenčným motívom v deskriptore, ako aj pozíciu hrán rprezentujúcich interakcie medzi segmentami. Takéto zarovnanie deskriptora k vstupnej sekvencii popíšeme štyrmi sadami binárnych premenných:

- **Premenné pre umiestnenie segmentov** x_{ij}: premenná x_{ij} má hodnotou 1 práve vtedy, keď na pozícii i v sekvencii je zarovnaný j-ty segment deskriptora. Premenných tohto typu je $N \times S$, kde N je dĺžka vstupnej sekvencie a S je počet segmentov v deskriptore.

- **Premenné pre umiestnenie sekvenčných motívov** m_{ij}: premenná m_{ij} má hodnotu 1 práve vtedy keď, na pozícii i v sekvencii začína sekvenčný motív j-teho segmentu deskriptora. Pre každý sekvenčný motív M_j špecifikovaný v deskriptore, máme $N - length(M_j) + 1$ premenných typu m_{ij} (motív musí byť zarovnaný na sekvencii celou dĺžkou).

- **Premenné pre spárované segmenty** y_{ijkl}, z_{ijkl}: premenná y_{ijkl} má hodnotu 1 práve vtedy, keď medzi segmentami j a l je paralelná hrana, ktorá začína párom aminokyselín i (v j-tom segmente) a k (v l-tom segmente). Tieto hrany majú zvyčajne dĺžku väčšiu ako 1, to znamená, že pozostávajú z viacerých párov interagujúcich aminokyselín, ktoré sa na sekvencii posúvajú ob jednu aminokyselinu. Teda paralelná hrana dĺžky n medzi segmentami j a l začínajúca na aminokyselinách i a k predstavuje páry aminokyselín $(i, k), (i + 2, k + 2)...(i + 2j, k + 2j)...(i + 2(n - 1), k + 2(n - 1))$.

  Napríklad ak medzi segmentami j a l je paralelná hrana dĺžky 3, ktorá začína aminokyselinovým párom i a k, premenná $y_{ijkl} = 1$, ostatné (y_{*j*l}) premenné pre túto väzbu budú mať hodnotu nula. Keďže má hrana medzi segmentami j a l dĺžku 3, $y_{ijkl} = 1$ znamená, že spárované sú aj aminokyselinové páry $(i+2, k+2)$ a $(i+4, k+4)$.

  Podobne premenná z_{ijkl} má hodnotu 1 práve vtedy, keď medzi segmentami j a l je antiparalelná hrana, ktorá začína párom aminokyselín i (v j-tom segmente) a k (v l-tom segmente). V prípade antiparalelného smeru ďalšie páry tejto väzby spájajú aminokyseliny nasledujúce aminokyselinu i a predchádzajúce aminokyselinu k. Takže antiparalelná hrana dĺžky n medzi segmentami j a l začínajúca na aminokyselinách i a k predstavuje páry aminokyselín $(i, k), (i + 2, k - 2)...(i + 2j, k - 2j)...(i + 2(n - 1), k - 2(n - 1))$.

Pre čo najrýchlejšie dosiahnutie optimálneho riešenia potrebujeme obmedziť prehľadávaný priestor a teda čo možno najmenší počet premenných. Preto sa pri generovaní premenných pre spárované segmenty berú do úvahy podmienky, ktoré vyplývajú z obmedzení na dĺžku segmentov určených deskriptore. Prípustné hodnoty pre začiatočnú aminokyselinu i väzby medzi segmentami j a l ležia v intervale určenom súčtom minimálnych dĺžok segmentov 1 ... $j-1$ a súčtom maximálnych dĺžok segmentov 1 ... j, od ktorého je odpočítaná dĺžka väzby. Pre každú začiatočnú aminokyselinu väzby i z tohto intervalu sa rovnakým spôsobom sa dajú ohraničiť aj prípustné hodnoty pre druhú aminokyseliny v páre k patriacu do segmentu l. Navyše sa podobným výpočtom tieto prípustné hodnoty dajú ohraničiť aj zprava (súčet k a minimálnych dĺžok segmentov, ktoré nasledujú za segmentom l nesmie byť väčší ako je dĺžka vstupnej sekvencie).

- **Pomocné premenné pre kontrolu prekrývania väzieb** p_{ijl}: premenná p_{ijl} má hodnotu 1 práve vtedy, keď aminokyselina i vystupuje v niektorom z párov aminokyselín, ktoré tvoria hranu medzi segmentami j a l. Podmienky s týmito premennými zaručia, že jedna aminokyselina nebude vystupovať v dvoch rôznych väzbách.

Daných S segmentov explicitne zapísaných v deskriptore ešte rozšírime o špeciálne pomocné segmenty 0 a $S+1$, ktoré zodpovedajú aminokyselinám na vstupnej sekvencii, ktoré nie sú zarovnané s deskriptorom kódujúcim danú doménu.

Účelová funkcia binárneho programu vygenerovaného pre daný deskriptor a vstupnú sekvenciu má tvar:

$$\sum_{i,j}(e_{ij}x_{ij} + f_{ij}m_{ij}) + \sum_{i,j,k,l}(g_{ijkl}y_{ijkl} + h_{ijkl}z_{ijkl})$$

,

kde e_{ij} je skóre pre zhodu medzi očakávanou sekundárnou štruktúrou aminokyseliny i a sekundárnou štruktúrou segmentu j, m_{ij} je skóre zhody sekvenčného motívu segmentu j a sekvencie na úseku začínajúceho aminokyselinou i, g_{ijkl} (h_{ijkl}) je skóre paralelnej (antiparalelnej) hrany medzi segmentami j a l, ktorá začína spárovanými aminokyselinami i a k.

Skóre pre páry aminokyselín $(A_i - A_k, A_{i+2} - A_{k+2}, ..., A_{i+(2length(E_{jl})-1)} - A_{k+(2length(E_{jl})-1)})$ sa získa z natrénovaných SVM, ktoré sme popísali v kapitole 4. Koeficienty e_{ij}, f_{ij}, g_{ijkl} a h_{ijkl} sú vypočítané na základe skórovacej schémy z kapitoly 3.

Aby uvedené sady binárnych premenných kódovali prípustné zarovnanie deskriptora ku sekvencii, musia premenné zároveň spĺňať nasledujúce podmienky:

$$(S_1) \quad \forall i : \sum_j x_{ij} = 1$$

$$(S_2) \quad \forall i \geq 1, j \geq 1 : x_{ij} - x_{i-1,j} \leq x_{i-1,j-1}$$

$$(S_3) \quad \forall i \geq 1 : x_{i,0} \leq x_{i-1,0}$$

Podmienka S_1 zabezpečuje, aby každá aminokyselina patrila ku práve jednému segmentu.

Podmienka S_2 zabezpečuje, aby segmenty v zarovnaní nasledovali za sebou v správnom poradí. Inými slovami, ak aminokyselina i patrí do j-teho segmentu ($x_{ij} = 1$), tak buď aj predchádzajúca aminokyselina i patrí do toho istého segmentu j ($x_{i-1,j} = 1$), alebo patrí do predchádzajúceho segmentu j-1 ($x_{i-1,j-1} = 1$). Takže dve za sebou idúce aminokyseliny sú buď v zasebou idúcich segmentoch alebo tom istom segmente.

Podmienka S_3 ošetruje hraničné prípady podmienky S_2, konkrétne zabezpečuje, aby bol prvý segment súvislý.

$$(L_1) \quad \forall j : \text{lower}_j \leq \sum_i x_{ij} \leq \text{upper}_j$$

Hodnoty lower_j a upper_j označujú minimálnu a maximálnu dĺžku segmentu určenú v deskriptore. Podmienka L_1 kontroluje, či počet aminokyselín priradených segmentu j spadá do tohto intervalu.

$$(M_1) \quad \forall j : \sum_i m_{ij} = 1$$

$$(M_2) \quad \forall i, j : m_{ij} \leq x_{ij}$$

$$(M_3) \quad \forall i, j : m_{ij} \leq x_{i+\text{motlen}_j-1,j}$$

Podmienka M_1 zabezpečuje, aby mal každý sekvenčný motív, ktorý je špecifikovaný v deskriptore, priradenú k práve jednu začiatočnú aminokyselinu.

Podmienky M_2 a M_3 kontrolujú, či aminokyseliny, na ktorých je zarovnaný motív segmentu j, patria do správneho segmentu v zarovnaní. Podmienka M_2 takto ošetruje začiatok motívu, teda ak je $m_{ij} = 1$, musí platiť aj $x_{ij} = 1$. . Výraz motlen_j vyjadruje dĺžku motívu špecifikovaného v deskriptore pre j-ty segment a index $i + \text{motlen}_j - 1$ popisuje poslednú aminokyselinu v motíve. Podmienka M_3 zabezpečuje, že aj táto aminokyselina patrí do segmentu j, a teda z podmienok M_3 a S_2 vyplýva, že všetky aminokyseliny priradené k motívu sú v správnom segmente.

$$(P_1) \quad \forall i, j, k, l, 0 \leq \ell < b_{jl} : y_{ijkl} \leq x_{i+2\ell,j}$$

$$(P_2) \quad \forall i, j, k, l, 0 \leq \ell < b_{jl} : y_{ijkl} \leq x_{k+2\ell,l}$$

$$(P_3) \quad \forall j, l : \sum_{i,k} y_{ijkl} = 1$$

Hodnota b_{jl} označuje dĺžku hrany medzi segmentami j a l špecifikovanú v deskriptore. Podmienka P_1 zabezpečuje, že začiatočné aminokyseliny z interagujúcich párov, ktoré sú súčasťou tejto paralelnej hrany, patria do segmentu j.

Podmienka P_2 zabezpečuje, že končiace aminokyseliny z interagujúcich párov, ktoré sú súčasťou tejto paralelnej hrany, patria do segmentu l.

Podmienka P_3 zabezpečuje, že každá paralelná hrana špecifikovaná deskriptorom má určený začiatok v zarovnaní.

$$(A_1) \quad \forall i, j, k, l, 0 \leq \ell < b_{jl} : z_{ijkl} \leq x_{i+2\ell,j}$$

$$(A_2) \quad \forall i, j, k, l, 0 \leq \ell < b_{jl} : y_{ijkl} \leq x_{k-2\ell,l}$$

$$(A_3) \quad \forall j, l : \sum_{i,k} y_{ijkl} = 1$$

Podmienka A_1 zabezpečuje, že začiatočné aminokyseliny z interagujúcich párov, ktoré sú súčasťou tejto antiparalelnej hrany, patria do segmentu j.

Podmienka A_2 zabezpečuje, že končiace aminokyseliny z interagujúcich párov, ktoré sú súčasťou tejto antiparalelnej hrany, patria do segmentu l

Podmienka A_3 zabezpečuje, že každá antiparalelná hrana špecifikovaná deskriptorom má určený začiatok v zarovnaní.

$$(H_1) \quad \forall i,j,l : p_{ijl} = \sum_{\ell=0}^{b_{jl}-1} \sum_{k} y_{i-2\ell,j,k,l}$$

$$(H_2) \quad \forall i,j,l : p_{ijl} = \sum_{\ell=0}^{b_{jl}-1} \sum_{k} z_{i-2\ell,j,k,l}$$

$$(H_3) \quad \forall i : \sum_{j,l} p_{ijl} \leq 1$$

Podmienky H_1 a H_2 vynútia, aby pomocné premenné p_{ijl} mali hodnotu 1, práve vtedy keď aminokyselina i vystupuje vo väzbe medzi segmentami j a l.

Podmienka H_3 kontroluje, či sú všetky aminokyseliny použité najviac v jednom interagujúcom páre.

Tieto podmienky zabezpečujú, že priradenie hodnôt 0 alebo 1 ku sadám premenných zodpovedá zarovnaniu deskriptora k vstupnej sekvencii. Maximalizovaním účelovej funkcie programu dostaneme optimálne zarovnanie vzhľadom na našu skórovaciu schému.

5.3 Použitie celočíselného programu reálnych dátach

Vyššie uvedený celočíselný program sme použili nariešenie problému zarovnania deskriptora OB-fold domény ku sade proteínových sekvencií, ktorá obsahovala 10 proteínov s OB-fold doménou a 13 náhodne vybraných proteínov bez tejto domény. Tieto sekvencie sme vybrali z databázy SwissProt (Bairoch et al., 2004). Pre účely testovania celočíselného programu sme z proteínov vybrali úseky sekvencie obsahujúce OB-fold doménu s dĺžkou menšou než 200 (pre negatívne proteíny boli použité náhodné úseky danej dĺžky). Pomocou nástroja PSIPRED (Jones, 1999) sme získali predikciou príslušnosti k jednotlivým typom sekundárnej štruktúry pre aminokyseliny zo sekvencie.

Pre vygenerovanie celočíselného programu pre daný deskriptor a vstupnú sekvenciu sme vytvorili program v jazyku *Java*, ktorý z deskriptora pre OB-fold domény, sekvencie aminokyselín s predikovanými pravdepodobnostami pre jednotlivé typy sekundárnej štruktúry a natrénovaných SVM modelov vygeneroval podmienky k celočiselnému programu pre

protein	dĺžka	P/N	skóre	čas
domain_A2QSY5	155	N	12.863	3d 14h 42m 39s
domain_Q9MUM1	155	N	12.2327	18h 20m 9s
domain_C6DDH0	155	N	-1.0506*	>18d
domain_P23180	155	N	6.2340	16d 21h 29s
domain_B0UU36	155	N	20.1410	6h 55m 27s
domain_B7LAR7	155	N	6.7946	1d 14h 6m 59s
domain_Q230X8	155	N	2.6504	3d 2h 57m 24s
domain_Q2YMQ2	155	N	-1.8042	1d 12h 34m 9s
domain_Q5M1N8	120	N	3.5389	1d 8h 6m 43s
domain_Q72U22	155	N	-0.6774*	>18d
domain_B0K889	155	N	21.7818	15d 1h 34m 41s
domain_Q90229	155	N	8.3781	18d 22h 17m 3s
domain_Q5A455	155	N	29.9385	1h 21s
domain_TEBH_EUPCR	155	P	44.3315	32m 55s
domain_POT1_SCHPO	146	P	39.355	36m 41s
domain_POTE1_CHICK	131	P	44.7688	4m 18s
domain_POTE1_HUMAN	131	P	45.4261	3m 12s
domain_POTE1_MOUSE	131	P	43.6156	5m 42s
domain_TEB EUPCR	120	P	47.3266	47s
domain_TEBA_OXYNO	160	P	51.1779	22m 43s
domain_TEBA_STYMY	160	P	51.2706	29m 47s
domain_CDC13_YEAST	195	P	54.7663	2m 12s
domain_POTE1_MACFA	131	P	45.2819	2m 47s

Tabuľka 5.1: Výsledky pre hľadanie zarovnania pomocou ILP. Stĺpec P/N vyjadruje, či bola skúmaná doména pozitívnym príkladom alebo nie. V niektorých prípadoch sa výpočet neskončil za väčší počet dní. Znak * pri skóre indikuje najlepšie nájdené celočíselné riešenie za daný čas.

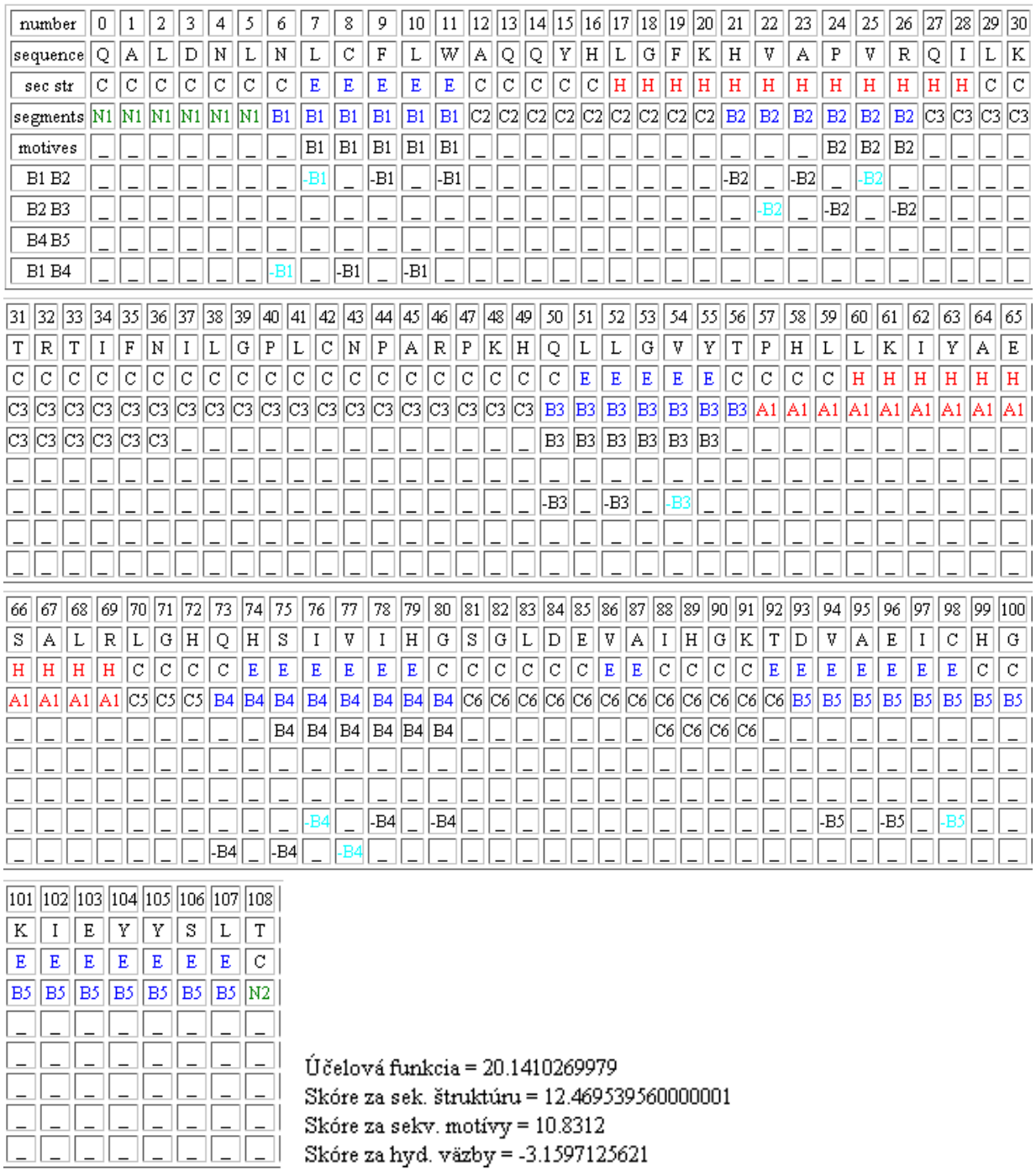
zarovnanie a vypočítal koeficienty v účelovej funkcii. Následne sme takto vygenerovaný celočíselný program riešili solverom CPLEX.

Na obrázkoch 5.1 a 5.2 sú príklady zarovnania vypočítaného naším celočíselným lineárnym programom. Sekvencia aminokyselín, ku ktorej zarovnávame deskriptor, je v druhom riadku zarovnania. V treťom je predpokladaná sekundárna štruktúra na danej aminokyseline, predikovaná pomocou nástroja PSI-Pred. Vo štvrtom riadku sú vyznačené pozície, na ktoré boli zarovnané jednotlivé segmenty deskriptora. Piaty riadok obsahuje umiestnenie sekvenčných motívov, ktoré boli v deskriptore špecifikované pre niektoré segmenty. Zvyšné

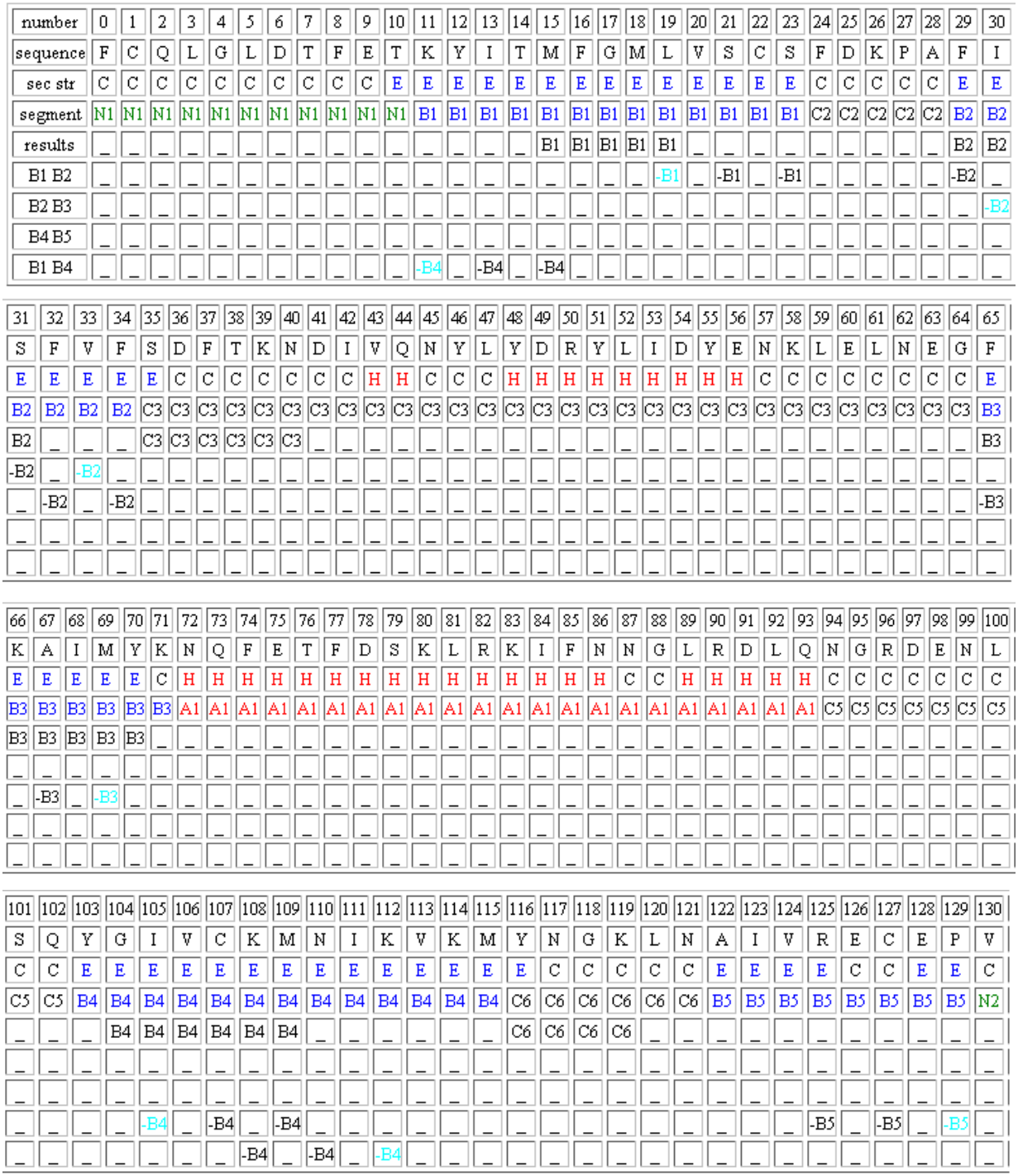
štyri riadky popisujú interagujúce aminokyseliny pre každú zo štyroch hydrogénových väzieb popísaných v deskriptore.

Výsledky pre najlepšie zarovnanie sú v tabuľke 5.1 a rozdiel medzi skóre, ktoré dosiahli pozitívne a negatívne príklady je dostatočne veľký na korektnú separáciu pozitívnych a negatívnych príkladov. Avšak časová náročnosť výpočtu je veľmi veľká, obzvlášť pri negatívnych príkladoch.

Keďže na tento problém narazíme už pri testoch na krátkych úsekoch proteínov, dlhý čas výpočtu znemožňuje praktické použitie celočíselného programovania na hľadanie OB-fold domény v celých proteínoch. Z tohto dôvodu musíme nájsť výrazne rýchlejšiu metódu na určovanie najlepšieho zarovnania, ktorý popíšeme v nasledujúcej kapitole.



Obr. 5.1: Príklad vypočítaného zarovnania pre úsek proteínu B0UU36. Tento proteín neobsahuje doménu Telo_bind, ktorú hľadáme deskriptorom a dosiahol skóre 20.141. V zarovnaní vidíme, že segment $B2$ bol zarovnaný na úsek sekvencie, kde bol predikovaný α-helix. Zarovnanie potom dosiahlo záporné skóre za väzby medzi β-listami.



Obr. 5.2: Príklad vypočítaného zarovnania pre úsek proteínu CDC13. Tento proteín obsahuje doménu Telo_bind, ktorú hľadáme deskriptorom a dosiahol celkové skóre 54.7663.

Kapitola 6

Zarovnávanie deskriptorov pomocou dynamického programovania

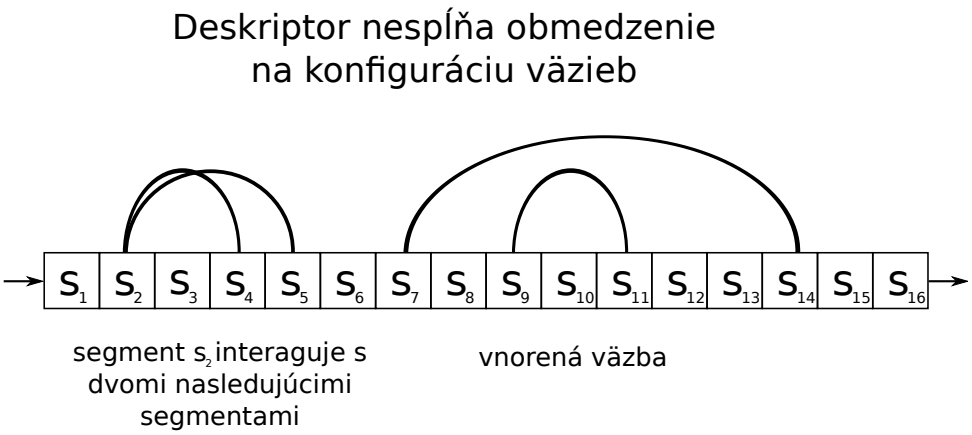
6.1 Relaxovaný problém

Všeobecne formulovaný problém zarovnania deskriptoru ku sekvencii môže popisovať ľubovoľné interakcie medzi vzdialenými segmentami, pričom pozície spárovaných aminokyselín nemajú žiadne ďalšie obmedzenia. Táto voľnosť spôsobuje, že problém, ktorého riešenie hľadáme, je podobný NP-ťažkým problémom, ako je to napríklad v prípade zarovnávania RNA deskriptoru s pseudouzlami ku sekvencii (Rampasek, 2011). Riešenie všeobecne formulovaného problému teda môže byť časovo veľmi náročné.

Ak zavedieme ďalšie obmedzenia na konfiguráciu interakcií medzi segmentami popísanými v deskriptore, je možné využiť toto zúženie problému a nájsť rýchlejšie riešenie pre určitú množinu deskriptorov.

Ak medzi dvoma segmentami s_1 a s_2 existuje v deskriptore hydrogénová väzba, budeme túto skutočnosť označovať $s_1 \sim s_2$. Ak segment s_1 leží v deskriptore s_2, označujeme túto skutočnosť ako $s_1 < s_2$. Segment s je *osamelý*, ak nevystupuje v žiadnej hydrogénovej väzbe.

Definícia 6.1. Deskriptor spĺňa *obmedzenie na konfiguráciu väzieb* ak pre každé dva segmenty $s_1 < s_2$, pre ktoré $s_1 \sim s_2$, platí, že všetky segmenty s ležiace medzi s_1 a s_2 ($s_1 < s < s_2$) sú osamelé.



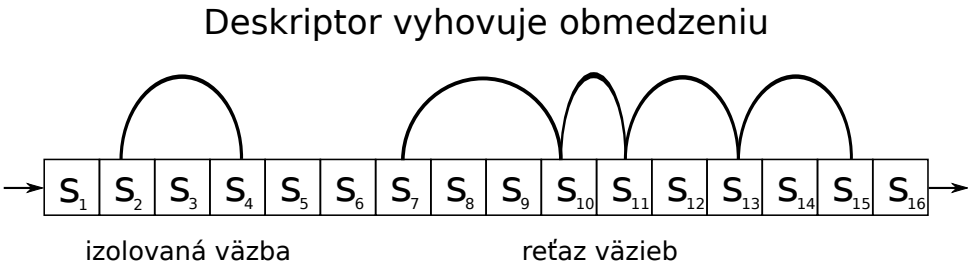
Obr. 6.1: Deskriptor porušuje podmienku, pretože segment s_2 interaguje so segmentami s_4 a s_5. Podľa obmedzenia z definície 6.1 nesmú segmenty medzi s_2 a s_5 spojenými väzbou ($s_2 \sim s_5$) vystupovať v ďalšej väzbe. Toto je porušené väzbou ($s_2 \sim s_4$). Rovnako toto obmedzenie porušujú interagujúce segmenty s_9 a s_11, ktoré ležia medzi segmentami interagujúcimi segmentami s_7 a s_14.

Príklady konfigurácie väzieb, ktoré porušujú toto obmedzenie, sú na obrázku 6.1. Deskriptory, ktoré uvažujeme v relaxovanom probléme zarovnania, spĺňajú obmedzenie z definície 6.1 a teda vyjadrujú len jednoduchšiu štruktúru väzieb medzi segmentami. Táto vlastnosť deskriptorov zaručí, že každý segment interaguje s najviac dvoma inými segmentami (s jedným segmentom pred ním a jedným za ním). Zároveň zabezpečí, že deskriptor nebude obsahovať vnorené väzby, teda napríklad segmenty s_1, s_2, s_3, s_4 nemôžu interagovať v pároch $s_1 \sim s_4$ a $s_2 \sim s_3$.

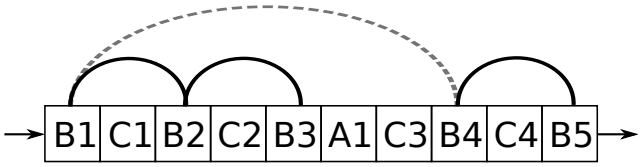
Väzby v deskriptoroch, ktoré spĺňajú obmedzenie z definície 6.1, sú buď izolované alebo môžu na seba nadväzovať a vytvárať tak jednoduché štruktúry.

Definícia 6.2. Postupnosť aspoň dvoch väzieb $(s_1 \sim s_2), (s_2 \sim s_3)..(s_{k-1} \sim s_k)$, v ktorých koncový segment jednej väzby je začiatočným segmentom nasledujúcej väzby, nazveme *reťaz väzieb*.

Takouto reťazou väzieb je napríklad postupnosť väzieb $(s_7 \sim s_{10})$, $(s_{10} \sim s_{11})$, $(s_{11} \sim s_{13})$ a $(s_{13} \sim s_{15})$ v deskriptore na obrázku 6.2. Tieto reťaze tvoria menšie navzájom nezávislé oblasti deskriptora, čo umožňuje riešiť problém zarovnania deskriptora pomocou dynamického programovania.



Obr. 6.2: Deskriptor spĺňa obmedzenie z definície 6.1, pretože obsahuje len izolovanú väzbu a reťaz väzieb z definície 6.2



Obr. 6.3: Prerušovanou čiarou je označená väzba medzi segmentami $B1$ a $B4$, ktorú z deskriptora pre Telo_bind doménu vynecháme. Takto upravený deskriptor spĺňa obmedzenie z definície 6.1.

Deskriptor pre OB-fold domény popísaný v kapitole 3 (obrázok 3.7) nespĺňa obmedzenie z definície 6.1, pretože väzba $B1 \sim B4$ pokrýva väzbu $B2 \sim B3$ a má spoločný začiatok s väzbou $B1 \sim B2$. Z tohto dôvodu budeme v dynamickom programovaní používať deskriptor, ktorý neobsahuje túto väzbu (viď. obrázok 6.3). Po určení najlepšieho zarovnania pre tento zjednodušený deskriptor do výsledného zarovnania dopočítame umiestnenie preskočenej väzby $B1 \sim B4$, pričom vyberieme najlepšie ohodnotené miesto v sekvencii pre začiatok väzby v segmentoch $B1$ a $B4$ fixovaných podľa zarovnania zo zjednodušeného deskriptora. Týmto spôsobom dostaneme odhad optimálneho riešenia pre kompletný deskriptor.

6.2 Dynamické programovanie

Najlepšie zarovnanie zjednodušeného deskriptoru k danej sekvencii dokážeme vypočítať v polynomiálnom čase prostredníctvom dynamického programovania. V tejto kapitole uvedieme postupne niekoľko variantov dynamického programovania, pričom každý ďalší variant rieši zložitejšiu a všeobecnejšiu verziu problému.

6.2.1 Zarovnávanie bez väzieb medzi segmentami

Základný variant rozoberá situáciu, keď sú v deskriptore špecifikované len segmenty a ich sekvenčné motívy, ale žiadne väzby medzi jednotlivými segmentami. Dynamickým programovaním budeme riešiť problém, ako dostať najlepšie zarovnanie pre prvých n segmentov deskriptora, za predpokladu, že poznáme najlepšie zarovnanie pre prvých $n-1$ segmentov. Najlepšie skóre pre zarovnanie prvých n segmentov deskriptora, kde n-tý segment končí na k-tej pozícii v sekvencii, označíme $A[n,k]$. Toto skóre získame z rekurentného vzťahu:

$$A[n,k] = max_f(A[n-1,f-1] + S[n,f,k]),$$

kde $S[n,f,k]$ je skóre n-tého segmentu začínajúceho na pozícii f ($f < k$) a končiaceho na pozícii k. V prípade, že má segment n špecifikovaný aj sekvenčný motív, skóre $S[n,f,k]$ zahŕňa aj najlepšie ohodnotené umiestnenie tohto motívu medzi aminokyselinami f a k. Táto rekurencia vyjadruje, že najlepšie zarovnanie pre n segmentov získame, tak že určíme najvhodnejšie miesto začiatku n-tého segmentu f. Túto pozíciu vyberieme tak, aby sme dosiahli maximum pri súčte skóre za zarovnanie $n-1$ segmentov, ktoré končí na pozícii $f-1$ a skóre za zarovnanie n-tého segmentu ku sekvencii od pozície f po pozíciu k.

Základný prípad v tomto rekurentnom vzťahu je určenie skóre pre zarovnanie prvého segmentu v deskriptore končiaceho na danej pozícii k. Tento základný prípad sa vypočita priamočiaro:

$$A[1,k] = max_f(S[1,f,k]),$$

Pri počítaní zarovnania najskôr vypočítame skóre zarovnania prvého segmentu $A[1,k]$ pre všetky prípustné koncové pozície k a následne pomocou týchto hodnôt dokážeme spočítať $A[2,k]$ podľa vyššie uvedeného rekurentného vzťahu. Takýmto spôsobom dokážeme postupne vypočitať hodnoty matice A pre všetkých M segmentov deskriptora. $A[M,k]$ vyhodnotíme na každej prípustnej aminokyseline k, kde by mohol končiť posledný segment deskriptora M a maximum z týchto hodnôt predstavuje najväčšie skóre pre zarovnanie celého deskriptora. Konkrétne pozície segmentov a sekvenčných motívov v najlepšom zarovnaní vieme rekonštruovať na základe pozícií f, ktoré boli v priebehu výpočtu vybrané

pre jednotlivé segmenty.

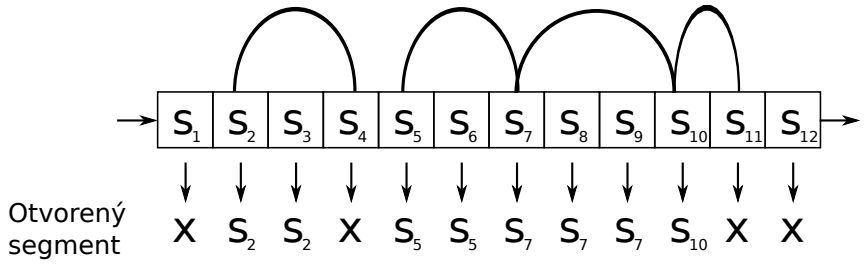
Maticu dynamického programovania dokážeme vypočítať v čase $O(nml^2)$, kde n je dĺžka vstupnej sekvencie aminokyselín, m je počet segmentov v deskriptore a l je dĺžka najdlhšieho segmentu v deskriptore. Túto zložitosť dostaneme, pretože rekurenciu počítame pre najviac n koncových aminokyselín, pričom sa m-krát snažíme umiestniť segment a pre každý segment skúšame l pozícií pre jeho začiatok a l pozícií pre začiatok sekvenčného motívu. Pamäťová zložitosť algoritmu je $O(nm)$, kde n je dĺžka sekvencie aminokyselín a m je počet segmentov v deskriptore.

6.2.2 Jedna aminokyselina môže patriť do viacerých väzieb

V prípade, že deskriptor obsahuje väzby medzi segmentami, skóre zarovnania musí zahŕňať ohodnotenie týchto hrán a preto potrebujeme popísané dynamické programovanie rozšíriť o túto zložku skóre. Túto úlohu najskôr zjednodušíme tým, že povolíme, aby jedna aminokyselina vystupovala naraz v dvoch rôznych väzbách. Táto formulácia nie je z biologického hľadiska realistická, pretože v interagujúcich β-listoch, ktoré popisujeme týmito väzbami, sa aminokyseliny viažu striedavo do zipsového vzoru (viď. obrázok 3.5). Z tohto dôvodu sa budeme neskôr venovať aj odstráneniu tohto zjednodušenia.

Nech výraz $B[n, f, k, f', k']$ vyjadruje najlepšie skóre, ktoré môže dosiahnuť hrana medzi segmentami n a n' ($n' < n$) v prípade, že segment n je zarovnaný medzi aminokyselinami f a k a segment n' medzi aminokyselinami f' a k'. Hrana medzi segmentami je definovaná prvým interagujúcim párom aminokyselín, pričom prvá aminokyselina v páre musí byť z intervalu (f', k') a druhá z (f, k). Na výpočet hodnoty $B[n, f, k, f', k']$ je potrebné nájsť začiatočný pár aminokyselín v týchto intervaloch, pre ktorý bude mať hrana maximálne skóre.

Keďže pre výpočet výrazu $B[n, f, k, f', k']$ využívame informáciu o rozsahu oboch segmentov, musíme si v matici dynamického programovania A uchovávať aj informáciu o pozícii segmentu n', kde začína väzba končiaca v segmente n. Pre popísanie segmentu, z ktorého vychádza väzba, zavedieme pojem *otvorený segment*.



Obr. 6.4: Segmenty, v ktorých väzba začína alebo sú medzi dvoma interagujúcimi segmentami, majú otvorený segment. Segmenty, nemajú otvorený segment a sú označené x.

Definícia 6.3. Nech $s_1 \sim s_2$. Potom segment s_1 je *otvoreným segmentom* pre každý segment s, pre ktorý platí $(s_1 \leq s < s_2)$.

Takže ak je väzba medzi dvoma segmentami s_1 a s_2, pre všetky ďalšie segmenty medzi týmito dvoma segmentami je začiatočný segment väzby s_1 ich otvoreným segmentom. Zároveň je segment s_1 aj svojím vlastným otvoreným segmentom. Ak segment neleží medzi dvoma segmentami tvoriacimi väzbu a ani v ňom nezačína žiadna väzba, ktorá by ho spájala so segmentom s väčším indexom, potom takýto segment nemá žiadny otvorený segment. Takýto segment môže vystupovať len vo väzbe, ktorá začína v segmente s nižším indexom. Príklad otvorených segmentov pre izlovanú hranu a reťaz väzieb je na obrázku 6.4.

Zo zjednodušujúceho obmedzenia z definície 6.1 vyplýva, že každý segment môže mať najviac jeden otvorený segment. Vďaka tomuto predpokladu môžeme pre výpočet zarovnania obsahujúceho aj skóre pre väzby v deskriptore rozšíriť pôvodnú maticu dynamického programovania $A[n, k]$ o hranice otvoreného segmentu f' a k' pre segment n. Takže nová matica $A[n, k, f', k']$ má prvky, ktoré určujú najlepšie skóre zarovnania prvých n segmentov, pričom segment n končí na pozícii k a jeho otvorený segment je zarovnaný medzi pozíciami f' a k'. Ak n nemá otvorený segment, na hodnotách parametrov f' a k' nezáleží, pretože hodnota prvkov matice $A[n, k, f', k']$ závisí len od parametrov n a k. V takom prípade budeme parametre f' a k' označovať symbolom $\perp$.

Rekurencia pre výpočet tejto matice bude závisieť od väzieb, ktoré sú pre segment n špecifikované v deskriptore. Vďaka podmienke, ktorá obmedzuje prípustné väzby v deskriptore, dokážeme určiť štyri možné prípady pre väzby interagujúce so segmentom n. Majme

segmenty $s_1 < n < s_2$. Rekurzívny výpočet matice $A[n, k, f', k']$ rozdelíme poďla týchto prípadov nasledovne:

- **Segment n neinteraguje so žiadnym segmentom.** Keďže v tomto prípade nemusíme riešiť skóre za interagujúcu väzbu, rekurencia sa podobá na problém zarovnania segmentov bez väzieb, ktorý sme riešili v predchádzajúcej časti.

  $$A[n, k, f', k'] = max_f(A[n-1, f-1, f', k'] + S[n, f, k])$$

  Skóre najlepšieho zarovnania pre n segmentov dostaneme maximalizáciou súčtu skóre pre najlepšie zarovnanie prvých $n-1$ segmentov a skóre za zarovnanie n-tého segmentu ku sekvencii na pozíciách f až k. Segment n môže mať otvorený segment, ak je medzi dvoma segmentami s_1 a s_2, ktoré sú spojené väzbou, a pre ktoré platí $s_1 < n < s_2$. V tomto prípade segment n má otvorený segment s_1, ktorého začiatok a koniec je f' a k'. Segment $n-1$ bude mať rovnaký otvorený segment a preto sa parametre f' a k' využijú pri výbere skóre zarovnania $n-1$ segmentov (výraz $A[n-1, f-1, f', k']$). Ak segment n nie je medzi segmentami spojenými hranou, a teda nemá otvorený segment, ani segment $n-1$ nebude mať otvorený segment. V tomto prípade tieto parametre neovplyvňujú hodnoty najlepšieho zarovnania pre $n-1$ ani pre n segmentov a teda hodnota f' aj k' je $\perp$.

- **Segment n interaguje len so segmentom s_2.** Tento prípad popisuje začiatok reťaze väzieb. Segment n začína väzbu ($n \sim s_2$), a teda je otvoreným segmentom sám sebe a jeho pozície f a k sú určené parametrami v matici $f' = f$ a $k' = k$. Keďže segment n interaguje len so segmentom s_2, segment $n-1$ nemá otvorený segment a neuvažujeme preň začiatok a koniec otvoreného segmentu ($A[n-1, f'-1, \perp, \perp]$). Skóre pre toto zarovnanie sa teda vypočíta priamočiaro:

  $$A[n, k, f', k'] = A[n-1, f'-1, \perp, \perp] + S[n, f', k']$$

  Keďže začiatočná pozícia n-tého segmentu f je v tomto prípade určená parametrom f', pri výpočte skóre najlepšieho zarovnania prvých n segmentov $A[n, k, f', k']$ nemusíme maximalizovať cez žiadne parametre.

- **Segment n interaguje so segmentom s_1 aj so segmentom s_2.** V tomto prípade segment n pokračuje v reťazi väzieb, končí väzbu $(s_1 \sim n)$ a teda uzatvára otvorený segment s_1. Preto sa do výrazu $A[n, k, f', k']$ započítava skóre $B[n, f', k', f'', k'']$ za väzbu $(s_1 \sim n)$. Zároveň segment n začína väzbu $(n \sim s_2)$ a je teda otvoreným segmentom sám sebe (ako aj segmentom $q : \ n \leq q < s_2$). Pretože n je sám pre seba otvorený segment, jeho začiatok f je určený hodnotami $f' = f$ a $k' = k$ z výrazu $A[n, k, f', k']$, takže sa tieto hodnoty použijú na výpočet skóre zarovnania n-tého segmentu ku sekvencii $S[n, f', k']$. Najlepšie zarovnanie pre prvých n segmentov sa teda v tomto prípade vypočíta nasledovne:

$$A[n, k, f', k'] = max_{f'', k''}(A[n-1, f'-1, f'', k''] + S[n, f', k'] + B[n, f', k', f'', k''])$$

Začiatočná a koncová pozícia n-tého segmentu (f a k) je určená hodnotami f' a k', preto hľadáme len maximálne parametre f'' a k'', ktoré určujú pozície segmentu s_1. Tento segment je otvoreným segmentom pre segment $n-1$ a takže jeho umiestnenie ovplyvňuje skóre za zarovnanie prvých $n-1$ segmentov $A[n-1, f'-1, f'', k'']$ a tiež skóre $B[n, f', k', f'', k'']$ za väzbu $(s_1 \sim n)$.

- **Segment n interaguje len so segmentom s_1.** Tento prípad predstavuje koniec reťaze väzieb. Keďže $s_1 < n$ segment n nemá otvorený segment, hodnoty f' a k' sú $\perp$. V tomto prípade musíme nájsť najlepšie skóre v závislosti od začiatočnej pozície n-tého segmentu f ako aj vzhľadom na končiacu väzbu $(s_1 \sim n)$. Všetky segmenty od s_1 po segment $n-1$ majú rovnaký otvorený segment, a to s_1, pre ktorý potrebujeme nájsť najvhodnejšie umiestnenie f'', k''. Preto musíme maximalizovať cez tri parametre:

$$A[n, k, f', k'] = max_{f, f'', k''}(A[n-1, f-1, f'', k''] + S[n, f, k] + B[n, f, k, f'', k''])$$

Do skóre sa teraz (okrem skóre za priloženie n-tého segmentu $S[n, f, k]$ a zarovnania prvých $n-1$ segmentov $A[n-1, f-1, f'', k'']$) pripočíta aj výraz $B[n, f, k, f'', k'']$, ktorý vyjadruje skóre pre najlepšie ohodnotenú väzbu $(s_1 \sim n)$. Táto väzba začína

v segmente s_1, ktorý je zarovnaný na pozíciách f'' až k''. Keďže s_1 je otvoreným segmentom aj pre segment $n-1$, určené pozície f'' a k'' segmentu s_1 sa musia použiť aj pri výpočte najlepšieho zarovnania prvých $n-1$ segmentov $A[n-1, f-1, f'', k'']$.

Základné prípady pre zarovnanie deskriptoru s väzbami je určenie skóre pre zarovnanie prvého segmentu v deskriptore končiaceho na danej pozícii k. Podľa toho či tento segment začína väzbu alebo nie rozlišujeme dva prípady:

- **Prvý segment nevystupuje v žiadnej väzbe.** V tomto prípade prvý segment nebude mať otvorený segment a hodnota jeho zarovnania nebude závisieť od parametrov f' a k', pre ktoré teda uvažujeme len hodnotu $\perp$. Skóre pre zarovnanie prvého segmentu ku sekvencii dostaneme určením najlepšej pozície f pre jeho začiatok.

$$A[1, k, f', k'] = max_f(S[1, f, k])$$

- **Prvý segment začína väzbu ($1 \sim s_1$).** Teda prvý segment otvára väzbu a je svojím vlastným otvoreným segmentom. Z toho vyplýva, že parametre f' a k' určujú začiatočnú a koncovú pozíciu prvého segmentu ($f = f'$, $k = k'$). Skóre pre zarovnanie prvého segmentu ku sekvencii v tomto prípade dostaneme priamočiaro:

$$A[1, k, f', k'] = S[1, f', k']$$

Pre riešenie problému zarovnania deskriptoru ku sekvencii opäť vypĺňame maticu dynamického programovania A od najlepšieho zarovnania prvého segmentu pre všetky prípustné koncové pozície k. Postupne pomocou uvedených rekurentných vzťahov vypočítame hodnoty v matici A až po najlepšie zarovnanie pre všetky prípustné koncové pozície k posledného segmentu v deskriptore. Následne z vypočítanej matice vyberieme najvyššiu hodnotu a zrekonštruujeme zarovnanie, ktoré toto skóre dosiahlo.

6.2.3 Jedna aminokyselina vystupuje najviac v jednej väzbe

Riešenie problému zarovnania uvedené v predchádzajúcej časti vypočíta najlepšie zarovnanie aj pre deskriptor s väzbami, ale umožňuje jednej aminokyseline patriť do dvoch

väzieb. Ak chceme, aby výsledné zarovnania popisovali páry aminokyselín, ktoré sa striedajú „zipsovým“ spôsobom (viď kapitola 3, obrázok 3.5), potrebujeme maticu dynamického programovania ešte rozšíriť o informáciu o interagujúcich aminokyselinách v reťaziach väzieb.

Pre izolované segmenty alebo samostatné väzby nie je potrebná dodatočná informácia a rekurentné vzťahy sa nezmenia. K zmene dôjde pri popisovaní rekurentných vzťahov pre segmenty, ktoré sú súčasťou reťaze väzieb. V tomto prípade máme segmenty, v ktorých jedna väzba končí a iná začína. Keďže jedna aminokyselina môže vystupovať najviac v jednej väzbe, umiestnenie prvej väzby v segmente limituje možnosti na zarovnanie druhej väzby.

Majme segmenty $s_1 < s_2 < s_3$, ktoré sú viazané väzbami $(s_1 \sim s_2)$ a $(s_2 \sim s_3)$. Súčasťou skóre zarovnania A pre prvých s_3 segmentov je skóre za zarovnanie väzby $(s_2 \sim s_3)$. Keďže možnosti zarovnania takejto väzby sú obmedzené pozíciou väzby $(s_1 \sim s_2)$, musíme maticu dynamického programovania A rozšíriť o ďalší parameter. Parameter r bude v matici dynamického programovania $A[s_3, k, f', k', r]$ označovať koncovú pozíciu už zarovnanej väzby $(s_1 \sim s_2)$ v segmente s_2, ktorý je otvoreným segmentom pre segment s_3. Z parametru r a z vlastností väzby $(s_1 \sim s_2)$, ktoré sú špecifikované v deskriptore (dĺžka, smer), vieme vypočítať, ktoré aminokyseliny nemôžu patriť do novej väzby $(s_2 \sim s_3)$. Informácie o väzbách v deskriptore sa využijú pri výpočte skóre pre zarovnanie väzby, ktorej umiestnenie je obmedzené už obsadenými pozíciami.

Skóre zarovnania takejto väzby označíme výrazom $BR[s_3, f, k, f', k', r]$. V tomto prípade výraz určuje skóre zarovnania väzby $(s_2 \sim s_3)$ medzi segmentami s_2 (na pozíciách f' až k') a s_3 (na pozíciách f až k), pričom toto zarovnanie je obmedzené aminokyselinami obsadenými umiestnením väzby $(s_1 \sim s_2)$ na pozícii r v segmente s_2.

Pri hľadaní zarovnania s takto obmedzenými väzbami musíme ošetriť dva nové prípady, ktoré vzniknú rozlišovaním izolovaných väzieb a väzieb s obmedzeniami na pozície v reťaziach väzieb. Výpočet matice dynamického programovania $A[n, k, f', k', r]$ pre kompletný problém zarovnania deskriptoru ku sekvencii je teda rozdelený na šesť prípadov. Majme segmenty $s_0 < s_1 < n < s_2$. Prvé tri prípady, ktoré ošetrujú segmenty bez väzieb a izolované

väzby, sa spracovávajú rovnako, ako v predchádzajúcom dynamickom programovaní:

- **Segment n neinteraguje so žiadnym segmentom.** Riešenie tohto prípadu je identické ako v dynamickom programovaní, ktoré nekontrolovalo aminokyseliny vybrané do väzieb.

  $$A[n, k, f', k', r] = max_f(A[n-1, f-1, f', k', r] + S[n, f, k])$$

  Skóre zarovnania pre prvých n segmentov sa vypočíta maximalizáciou súčtu skóre zarovnania pre prvých $n-1$ segmentov a skóre zarovnania n-tého segmentu ku sekvencii.

- **Segment n interaguje len so segmentom s_1, ktorý nemá ďalšiu väzbu.** Segment n končí izolovanú väzbu ($s_1 < n$), a teda nemá otvorený segment s obmedzenými pozíciami. Z tohto dôvodu sa parametre f', k' a r nevyužijú. Skóre sa opäť vypočíta rovnako, ako pri predchádzajúcom dynamickom programovaní.

  $$A[n, k, f', k', r] = max_{f, f'', k''}(A[n-1, f-1, f'', k'', r] + S[n, f, k] + B[n, f, k, f'', k''])$$

  K maximalizovanému súčtu sa pridá skóre zarovnania väzby bez obmedzených pozícií medzi segmentami n a s_1.

- **Segment n interaguje len so segmentom s_2, ktorý nemá ďalšiu väzbu.** V segmente n začína väzba ($n \sim s_2$) a tento segment nevystupuje v iných väzbách, výber pozície pre väzbu ($n \sim s_2$) teda nie je nijak obmedzený.

  $$A[n, k, f', k', r] = A[n-1, f'-1, \perp, \perp, \perp] + S[n, f', k']$$

  Rovnako ako pri predchádzajúcom dynamickom programovaní je v tomto prípade segment n svojím vlastným otvoreným segmentom, a preto sa skóre pre zarovnanie n-tého segmentu ku sekvencii $S[n, f, k]$ spočíta s parametrami f' a k' .

Zvyšné tri prípady pomocou parametra r riešia situáciu v reťazi väzieb:

- **Segment n je druhým segmentom v reťazi aspoň 2 väzieb ($s_0, s_1 \sim n \sim s_2$).**

  V tomto prípade sa skóre opäť počíta rovnako, ako pri predchádzajúcom dynamickom programovaní. Segment n končí väzbu ($s_1 \sim n$), ktorá nemá obmedzenie na pozície (v segmente s_1 žiadna väzba nekončí). Zároveň segment n začína väzbu ($n \sim s_2$), a teda je svojím vlastným otvoreným segmentom, čiže pozície segmentu určujú parametre f' a k'.

  $$A[n, k, f', k', r] = max_{f'',k''}(A[n-1, f'-1, f'', k'', r] + S[n, f', k'] + B[n, r, r, f'', k''])$$

  Parameter r určuje, na ktorej pozícii má byť väzba v otvorenom segmente pre segment n. Keďže n je svojím vlastným otvoreným segmentom, hodnota r určuje koncovú pozíciu uzatváranej väzby ($s_1 \sim n$), a preto sa skóre zarovnania tejto väzby vyjadrí výrazom $B[n, r, r, f'', k'']$. Súčet, ktorý dáva skóre zarovnania, sa maximalizuje nájdením najlepších pozícií f'' a k'' pre segment s_1, ktorý je otvoreným segmentom pre segment $n-1$.

- **Segment n je uprostred reťaze s aspoň dvoma väzbami ($s_1 \sim s_2 \sim n \sim s_3$).**

  Tento prípad je podobný predchádzajúcemu, ale väzba ($s_2 \sim n$), ktorú segment n uzatvára, má prípustné pozície obmedzené väzbou ($s_1 \sim s_2$).

  $$A[n, k, f', k', r] = max_{f'',k'',r''}(A[n-1, f'-1, f'', k'', r''] + S[n, f', k'] + BR[n, f', k', f'', k'', r"])$$

  Segment n je opäť otvoreným segmentom sám sebe, pretože otvára väzbu ($n \sim s_3$), a teda jeho pozície sú určené parametrami f' a k'. Keďže segment uzatvára väzbu ($s_2 \sim n$), započítava sa aj skóre za zarovnanie tejto väzby, ktoré je vyjadrené výrazom $BR[n, f', k', f'', k'', r"]$. V tomto prípade teda maximalizujeme súčet nielen cez pozície f'' a k'' segmentu s_2, ale aj v závislosti od pozície r'', ktorá určuje pozície obsadené väzbou ($s_1 \sim s_2$).

- **Segment n je posledným segmentom v reťazi aspoň dvoch väzieb ($s_1 \sim s_2 \sim n$).**

  Posledný prípad popisuje situáciu, keď segment n ukončuje reťaz aspoň dvoch väzieb. V tomto prípade väzba ($s_2 \sim n$), ktorú uzatvára segment n, má obmedzené pozície

väzbou ($s_1 \sim s_2$).

$$A[n, k, f', k', r] = max_{f,f'',k'',r''}(A[n{-}1, f'{-}1, f'', k'', r'']{+}S[n, f, k]{+}BR[n, f', k', f'', k'', r"])$$

Segment n už žiadnu ďalšiu väzbu neotvára, a teda nemá otvorený segment. Preto v tomto prípade súčet budeme maximalizovať aj cez parameter f, ktorý určuje začiatočnú pozíciu segmentu n. Ďalej budeme maximalizovať cez parametre f'' a k'', ktoré určujú pozíciu segmentu s_2, ako aj cez parameter r'', ktorý popisuje, ktoré pozície v tomto segmente sú obsadené väzbou ($s_1 \sim s_2$).

Základné prípady v tomto dynamickom programovaní sú rovnaké ako pre predchádzajúci variant, v ktorom sme dovolili viac väzieb na jednej pozícii. Skóre pre zarovnanie prvého segmentu v deskriptore končiaceho na danej pozícii k určíme podľa toho, či tento segment začína väzbu alebo nie:

- **Prvý segment nevystupuje v žiadnej väzbe.** Skóre pre zarovnanie prvého segmentu ku sekvencii dostaneme určením najlepšej pozície f pre jeho začiatok.

  $$A[1, k, f', k', r] = max_f(S[1, f, k])$$

  Keďže prvý segment nemá otvorený segment, hodnota jeho zarovnania nebude závisieť od parametrov f', k' a r. Pre tieto parametre teda uvažujeme len hodnotu $\perp$.

- **Prvý segment začína väzbu ($1 \sim s_1$).** Prvý segment otvára väzbu a je svojím vlastným otvoreným segmentom. Z toho vyplýva, že parametre f' a k' určujú začiatočnú a koncovú pozíciu prvého segmentu ($f = f'$, $k = k'$). Parameter r sa však nevyužije, pretože v prvom segmente nekončí žiadna iná väzba. Skóre pre zarovnanie prvého segmentu ku sekvencii v tomto prípade dostaneme priamočiaro:

  $$A[1, k, f', k', r] = S[1, f', k']$$

Aj v tomto prípade sa zarovnanie celého deskriptora vypočíta postupným vypĺňaním matice dynamického programovania A zo základných prípadov (najlepšie zarovnanie prvého

segmentu pre všetky prípustné koncové pozície k) až k zarovnaniam pre všetky segmenty v deskriptore končiace na daných pozíciách k. Z takto vypočítanej matice určíme najlepšie zarovnanie a jeho skóre.

6.2.4 Časová a pamäťová zložitosť dynamického programovania

Kompletný dynamický program pre problém zarovnania deskriptoru ku sekvencii pre zjednodušené deskriptory má časovú zložitosť $O(nmfl^4)$, kde n je dĺžka vstupnej sekvencie, m je počet segmentov, f je dĺžka maximálneho rozdielu medzi dvoma interagujúcimi segmentami a l je maximálna dĺžka segmentu v deskriptore. Dynamický program začneme na všetkých n prípustných aminokyselinách. V každej takejto vetve treba zarovnať m segmentov. Pri otvorení väzby treba určiť jej začiatočný segment, ktorý môže byť vo vzdialenosti f. Ďalej treba určiť začiatočný interagujúci pár tejto aminokyseliny, takže v každom segmente prehľadávame najviac l aminokyselín. Rovnako ako v dynamickom programe bez väzieb musíme určiť aj začiatok segmentu a začiatok jeho sekvenčného motívu. Z týchto dôvodov dostaneme v časovej zložitosti výraz l^4.

Pamäťová zložitosť tohto algoritmu je $O(nmfl^2)$ je určená rozmermi matice dynamického programovania. Matica má rozmery dané dĺžkou sekvencie n, počtom segmentov v deskriptore m, najväčším rozdielom medzi interagujúcimi segmentami f a najväčšou dĺžkou segmentu l, ktorá určuje možný rozsah pre parametre matice f' a r.

6.2.5 Poznámky k implementácii

Popísané dynamické programovanie umožňuje výpočítať zarovnania pre zjednodušené deskriptory v pomerne krátkom čase (rádovo minúty), avšak za predpokladu, že sú dopredu predpočítané skóre pre rôzne možnosti pozícií väzieb v deskriptore $B[n, f', k', f'', k'']$. Tieto hodnoty získavame pomocou klasifikácie cez SVM modely popísané v kapitole 4 pre všetky potenciálne interagujúce páry aminokyselín. Aby sme mohli ohodnotiť tieto interagujúce páry, musíme pre ne zo sekvencie vygenerovať vstupy pre SVM. Preto musíme dynamické programovanie počítať dvakrát. Pri prvom behu zistíme, ktoré úseky sekvencie musíme ohodnotiť pomocou SVM klasifikátora, aby sme mohli priradiť hodnoty pre všetky výrazy

$B[n, f', k', f'', k'']$. Pri tomto prechode vygenerujeme vstupy pre SVM klasifikátor, ale nie samotné zarovnanie. Až pri druhom prechode, keď vieme hodnotu každého výrazu $B[n, f', k', f'', k'']$, môžeme vypočítať najlepšie zarovnanie deskriptoru ku sekvencii (hodnoty v matici A).

Napríklad pre proteín CDC13, ktorého sekvencia je dlhá 924 aminokyselín, prvý beh dynamického programu (vygenerovanie vstupov pre SVM) trvá 1189 sekúnd a druhý beh (spočítanie najlepšieho zarovnania) trvá 143 sekúnd. Avšak vyhodnotenie 107599 vygenerovaných vstupov pre SVM (polynomiálny kernel s parametrom 7) trvá až 34908 (takmer 10 hodín) a toto ohodnotenie potenciálnych väzieb predstavuje časovo najnáročnejšiu časť výpočtu. Z tohto dôvodu výsledky pre jednotlivé vstupy ukladáme do hašovacej tabuľky. Takto ich nemusíme znova klasifikovať a môžeme priamo použiť už vypočítané skóre. Napriek tejto optimalizácii v prípade, že chceme deskriptor zarovnať s veľkým množstvom sekvencií, čas potrebný na ohodnotenie všetkých prípustných aminokyselinových párov je príliš veľký a preto potrebujeme obmedziť ich množstvo.

Aminokyseliny, ktoré nepatria do žiadneho β-listu, pravdepodobne nebudú súčasťou niektorej z väzieb medzi β-listami. Ak nebudeme vyhodnocovať páry, kde je aspoň jedna takáto aminokyselina, výrazne obmedzíme počet vstupov do SVM klasifikátora, a teda aj čas potrebný na výpočet zarovnania. Preto sme zaviedli orezávacie pravidlo, ktoré vyradí z vyhodnocovania SVM páry aminokyselín, kde aspoň jedna aminokyselina má predikovanú pravdepodobnosť príslušnosti do β-listu menšiu ako 50 %. Pre takéto aminokyseliny vrátime ako skóre hodnotu -∞. V prípade proteínov, ktoré nemajú dostatočne veľa aminokyselín s β sekundárnou štruktúrou nebude žiadna možnosť, ako zarovnať deskriptor ku sekvencii tak, aby väzby mali skóre väčšie ako -∞. Pre tieto proteíny teda neuvažujeme žiadne skóre zarovnania a predpokladáme, že proteíny s takýmito sekvenciami neobsahujú doménu popísanú deskriptorom. Toto riešenie môže spôsobiť, že budú orezané aj pozície, na ktorých by zarovnanie väzieb dosiahlo lepšie skóre než to, ktoré dostaneme po orezaní týchto pozícií. Avšak ako uvidíme vo výsledkoch experimentov, ktoré popíšeme v ďalšej časti, rozdiely medzi skóre zarovnania s orezávaním a bez neho sú zanedbatelné.

Hodnoty pre jednotlivé zložky skóre (sekundárna štruktúra, sekvenčné motívy a hydrogénové väzby) môžeme prenásobiť váhami a tak ovplyvniť mieru akou jednotlivé zložky

vystupujú v celkovom skóre. V našej práci sme zvolili váhu 1 pre skóre za štruktúru a motívy a váhu 2 pre hydrogénové väzby.

6.3 Experimentálne overenie metódy

6.3.1 Zostavenie testovacích množín

Klasifikačnú silu hľadania špecifických domén v proteínoch pomocou deskriptora sme overili na reálnych dátach. Dynamickým programovaním sme hľadali doménu Telo_bind z rodiny domén OB-fold, ktorej deskriptor je popísaný v kapitole 3. V prvom teste sme použili 23 proteínov (10 s hľadanou doménou a 13 bez nej), ktoré sme použili aj pri testovaní celočíselného lineárneho programovania v kapitole 5.

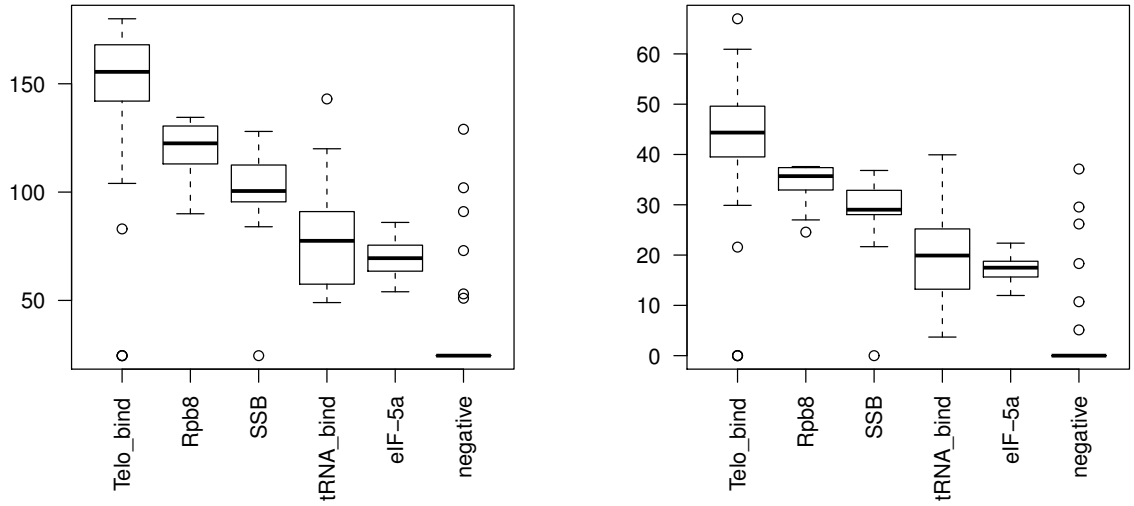
Pretože dynamické programovanie umožňuje v kratšom čase spočítať zarovnanie aj pre dlhšie sekvencie, zarovnávali sme deskriptor k celým proteínom a nie len k vybraným úsekom proteínu.

V tomto teste dva proteíny, ktoré neobsahujú OB-fold doménu, dosiahli veľmi dobré skóre. Ak sa pozrieme na výsledné zarovnanie v týchto prípadoch, vidíme, že skóre dosiahlo veľké hodnoty za sekundárnu štruktúru zarovnaním dlhých *coil* úsekov, čo prevážilo horšie skóre za β-listové segmenty a zle priložené profily a väzby. V ďalších analýzach budeme prihliadať k tejto možnosti. Na skóre ostatných zarovnaní opäť vidno dobrú separáciu negatívnych a pozitívnych príkladov výskytu hľadanej domény v proteíne. Skóre zarovnania s orezávaním pozícií, ktoré majú malú pravdepodobnosť príslušnosti k β-listu môžeme porovnať so skóre zarovnania bez tohto orezávania. Vidíme, že rozdiely medzi týmito skóre sú minimálne a v niektorých prípadoch obe metódy nájdu to isté zarovnanie.

Pri ďalšom teste sme z databázy Pfam (Finn et al., 2007) vybrali 50 proteínov, ktoré obsahujú hľadanú doménu Telo_bind a 50 proteínov bez nej. Následne sme z rovnakej databázy vybrali 20 proteínov pre štyri rodiny príbuzné k Telo_bind - rodina RNA polymerázy Rpb8 (PF03870 Rpb8), rodina proteínov viažúca jednovláknovú DNA (PF00436 SSB), rodina domén viažúcich tRNA (PF01588 tRna bind) a eukaryotický predlžovací faktor 5A hypusine (PF01287 elF-5a). Tieto domény patria do klanu OB-fold proteínov, ale

proteín	dĺžka	P/N	skóre bez orezávania	skóre s orezávaním
A2QSY5	281	N	18.8356	18.4966
Q9MUM1	446	N	13.3688	orezané
C6DDH0	302	N	17.5898	orezané
P23180	465	N	34.2771	orezané
B0UU36	334	N	20.6194	orezané
B7LAR7	200	N	15.5853	orezané
Q230X8	423	N	27.5368	27.1412
Q2YMQ2	435	N	5.4542	orezané
Q5M1N8	241	N	13.5218	orezané
Q72U22	307	N	7.2892	orezané
B0K889	233	N	21.3008	21.27
Q90229	280	N	61.1585	orezané
Q5A455	762	N	44.3158	44.3154
TEBH_EUPCR	460	P	45.9458	45.9458
POT1_SCHPO	555	P	44.3967	44.3967
POTE1_CHICK	778	P	55.247	55.247
POTE1_HUMAN	634	P	45.5979	45.5979
POTE1_MOUSE	640	P	42.3972	42.3972
TEB EUPCR	420	P	49.2403	49.2403
TEBA_OXYNO	495	P	52.2969	52.2398
TEBA_STYMY	493	P	52.5483	52.3672
CDC13_YEAST	924	P	58.9341	58.7209
POTE1_MACFA	634	P	46.7132	46.7132

Tabuľka 6.1: Tabuľka popisuje skóre najlepšieho zarovnania (s orezávaním a bez neho) deskriptoru pre Telo_bind doménu (viď kapitola 3, obrázok 3.8) ku sekvenciám 23 proteínov. Stĺpec P/N indikuje,či je daný proteín pozitívnym príkladom (P) a je v ňom prítomná doména popísaná deskriptorom alebo či daný proteín takúto doménu nemá (N).

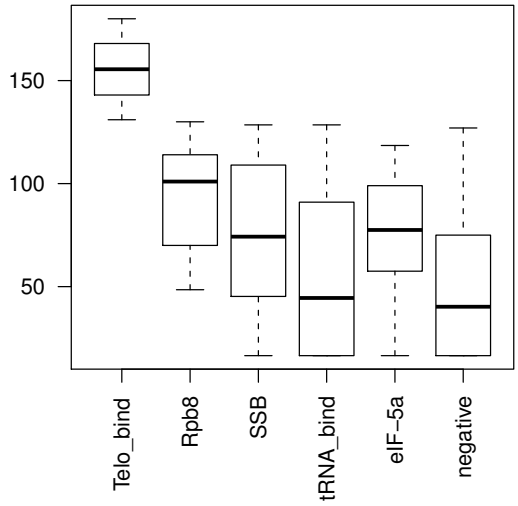


Obr. 6.5: Výsledky pre hľadanie Telo_bind deskriptor. Na osi x sú jednotlivé skupiny proteínov použité v testovacej množine. Na grafe vľavo sú na osi y je umiestnenie proteínov (rank) z danej skupiny vo výsledkoch hľadania. Na grafe v pravo je na osi y vyznačené skóre, ktoré dosiahli zarovnanie deskriptoru na jednotlivých proteínoch z danej skupiny.

nepatria do rodiny Telo_bind, v ktorej sú proteíny viažúce jednovláknový previs DNA sekvencie v telomérach. Takýmto spôsobom sme zostavili množinu 180 proteínov rôznej dĺžky, na ktorých sme deskriptorom hľadali Telo_bind doménu.

Výsledky tohto hľadania sú zobrazené na obrázku 6.5. Z výsledkov vyplýva, že táto metóda dokáže spoľahlivo separovať negatívne príklady od proteínov obsahujúcich OB-fold domény a zároveň dokáže odlíšiť špecifickú doménu Telo_bind od príbuzných rodín s OB-fold doménami. Najhoršie skóre pre pozitívny proteín bolo 21.585 a len tri negatívne príklady dosiahli vyššie skóre. Najlepšie ohodnotený proteín bez hľadanej domény dosiahol skóre 37.11, pričom len tri pozitívne príklady boli ohodnotené horšie. Tieto hodnoty nám poskytujú približnú predstavu o hranici medzi pozitívnymi a negatívnymi príkladmi. Zvolením určitého prahu medzi týmito hodnotami, dostaneme podmienku na klasifikáciu zarovnávaných sekvencií ako proteínov obsahujúcich Telo_bind doménu.

Proteíny s príbuznou doménou, ktorá nepatrila do rodiny Telo_bind, dosiahli hodnoty medzi 3.69 a 39.93. Tieto proteíny majú skóre blízko k Telo_bind proteínom, čo je výhoda,



Obr. 6.6: Výsledky pre hľadanie OB Fold domén pomocou Hmmeru. Na osi x sú jednotlivé skupiny proteínov použité v testovacej množine. Na osi y je umiestnenie proteínov (rank) z danej skupiny vo výsledkoch hľadania.

pretože to indikuje možnosť využitia deskriptora na hľadanie vzdialených ortológov a proteínov s príbuznou doménou.

6.3.2 Porovnanie s programom Hmmer

Klasifikačnú silu našej metódy sme porovnali s nástrojom Hmmer (Eddy, 2011), pomocou ktorého bola vytvorená databáza Pfam. Keďže sme testovacie sady vytvárali podľa príslušnosti proteínov k rodinám špecifikovaným v databáze Pfam, nie je prekvapivé, že nástroj Hmmer dosahuje dokonalú separáciu pozitívnych a negatívnych príkladov pre proteín obsahujúci Telo_bind doménu. V prípade OB-fold proteínov z rodín vzdialenejšie príbuzných k rodine Telo_bind však nástroj Hmmer nedetekuje podobnosť a vo výsledku sú tieto proteíny nerozlíšiteľné od negatívnych príkladov proteínov bez OB-fold domény. Toto poukazuje na výhodu nášho prístupu, ktorý dokáže od negatívnych príkladov

proteín	skóre za sek. štruktúru	skóre za sekv. motívy	skóre za hyd. väzby	celkové skóre
Q6CH82_YARLI	19.8597	17.8095	4.0039	41.6731
Q6C3B9_YARLI	24.8945	16.4571	1.8168	43.1684
Q6CEL3_YARLI	36.2416	19.5187	1.4999	57.2602
ATG2_YARLI	27.5304	18.6221	1.2247	47.3772
Q6C292_YARLI	26.8503	18.3205	1.0003	46.1711
Q6C1C4_YARLI	30.8243	11.3835	0.8913	43.0991
Q6C9Z7_YARLI	17.0983	20.798	0.4499	38.3462
Q6C7F9_YARLI	27.0412	12.8768	0.4373	40.3553
Q6CGL3_YARLI	25.6103	15.0445	0.0106	40.6654
Q6C419_YARLI	35.9560	13.0431	-0.1335	48.8656
Q6C463_YARLI	43.9581	12.8774	-0.1627	56.6728
RSE1_YARLI	20.6491	19.2394	-0.4076	39.4809
Q6CHS1_YARLI	32.7372	10.7436	-0.5016	42.9792
Q6C8X2_YARLI	20.5557	19.546	-0.5517	39.55

Tabuľka 6.2: Tabuľka popisuje jednotlivé zložky skóre vybraných 14 proteínov kvasinky *Yarrowia lipolytica*.

6.3.3 Celogenómové hľadanie OB-Fold proteínov

Ako ďalší test sme skúsili hľadať proteíny s Telo_bind doménov v proteóme kvasinky *Yarrowia lipolytica*. Táto kvasinka je evolučne vzdialenejšia od najpoužívanejšieho modelového organizmu *Saccharomyces cerevisiae*, a preto je v nej sťažené hľadanie ortológov štandardnými metódami, ako napríklad Hmmer.

Deskriptor pre OB-fold doménu sme zarovnali k 6292 proteínom z kvasinky *Yarrowia lipolytica*. Veľká väčšina (4656) z týchto proteínov bola orezaných vyššie popísanou podmienkou ako proteíny, na ktoré sa nedá deskriptor zarovnať. Pre zvyšných 1636 proteínov bolo vypočítané najlepšie zarovnanie, pričom najlepšie dosiahnuté skóre bolo 62.236 a najnižšie bolo -42.618. Z týchto proteínov sa v analýze zameriame na proteíny, ktorých skóre zarovnania je lepšie ako 38, čo zúži počet analyzovaných proteínov na 80.

Použitý deskriptor obsahuje segmenty typu *coil*, ktoré majú veľkú maximálnu veľkosť, pretože v Telo_bind proteínoch sa vzdialenosti medzi β-listami často veľmi líšia. Táto väčšia veľkosť *coil* segmentov môže spôsobiť, že proteín získa veľmi veľké skóre za sekundárnu štruktúru tak, že sa priloží *coil* segment na dlhý úsek *coil* aminokyselín. Tieto vysoko

hodnotené zarovnania nemusia zodpovedať nájdenej Telo_bind doméne a potrebujeme ich rozlíšiť od relevantnejších výsledkov. Z tohto dôvodu sme usporiadali výsledky podľa zložky skóre za hydrogénové väzby a sústredili sme sa na zarovnania, ktoré sú v prvej stovke najlepšie hodnotených proteínov podľa tejto zložky, a zároveň patria do množiny 80 proteínov so skóre viac ako 38. Týmto spôsobom sme dostali 14 proteínov, ktorých výsledky sú v tabuľke 6.2.

Najlepšie hodnotený proteín z nich *Q6CEL3_YARLI* má skóre 57.26 a podľa informácie z databázy UniProt viaže tento proteín DNA, čo zodpovedá očakávaniam, pretože viazanie jednovláknovej DNA je funkciou hľadanej domény Telo_bind. Druhý najlepšie hodnotený proteín *Q6C463_YARLI* so skóre 56.67 nemá v UniProte anotovanú funkciu. Podobne nemá špecifikovanú funkciu ani proteín *Q6CH82_YARLI* s najlepším skóre za hydrogénové väzby (4.0039), ako aj ďalšie analyzované proteíny. Tieto proteíny bez anotovanej funkcie sú vhodnými kandidátmi na ďalšiu analýzu ich biologickej funkcie, prípadne na overenie tejto funkcie ďalšími experimentami.

Výsledky tohto experimentu demonštrujú, že metóda hľadania štrukturálnych domén pomocou deskriptora je aplikovateľná na úrovni celého genómu. Táto metóda môže slúžiť ako filter pre veľké množstvo sekvencií, z ktorých určí proteíny vhodné na ďalšiu analýzu. Avšak napríklad výskyt proteínov s vysokým skóre za coil segmenty poukazuje na potrebu dodatočného spracovania výsledkov tejto metódy, prípadne prehodnotenie štruktúry skórovacej funkcie.

Záver

V tejto práci sme sa zaoberali spôsobmi identifikovania ortologických proteínov. Najrozšírenejšie metódy používané na hľadanie ortológov sa zameriavajú na sekvenčnú podobnosť medzi príbuznými proteínmi. Z identifikovaných príbuzných proteínov s rovnakou funkciou z viacerých organizmov je možné vytvoriť reprezentáciu proteínov (napríklad pomocou profilových skrytých Markovovských modelov), ktorá sa zameriava na zhodné časti sekvencií v príbuzných proteínov a abstrahuje od častí špecifických pre jednotlivé organizmy. Tieto metódy prinášajú pozitívne výsledky pre veľké množstvo proteínov, ale ortológy, ktorých sekvencia počas evolúcie výraznejšie divergovala, takýmito metódami nemusia byť nájdené. V týchto proteínoch je často pre zachovanie biologickej funkcie dôležitejšia štruktúra, a preto je medzi ortologickými proteínmi viac zachovaná podobnosť v štruktúre než sekvenčná podobnosť. Cieľom v tejto práci preto bolo vytvoriť metódu, ktorá umožní reprezentáciu a hľadanie domén proteínov so známou štruktúrou pomocou využitia kombinácie vlastností ich primárnej, sekundárnej a terciárnej štruktúry.

Na riešenie tohto problému sme definovali reprezentáciu domén pomocou ručne zostavených deskriptorov, ktoré kombinujú viaceré typy informácií o štruktúre popisovanej domény. Táto reprezentácia bola inšpirovaná analogickými deskriptormi pre štrukturálne motívy v RNA. Na základe výsledkov prípadovej štúdie hľadania ortológov v rôznych druhoch kvasiniek pre telomerické proteíny sme sa rozhodli pre hľadanie domény Telo_bind. Táto doména je členom rodiny OB-fold domén, ktoré majú spoločnú charakteristickú štruktúru. β-listy tejto domény sa v trojrozmernom priestore poskladajú do tvaru β-barelu. Pre túto doménu sme vytvorili deskriptor, ktorý charakterizuje sekvenčné motívy, očakávanú se-

kundárnu štruktúru tejto domény, a to aj hydrogénové väzby medzi jednotlivými β-listami v β-barely.

Pre hľadanie takto reprezentovanej domény bolo potrebné nájsť spôsob, ako ohodnotiť možnú prítomnosť domény v sekvenciách kandidátskych proteínov. Na určenie potenciálnych hydrogénových väzieb medzi vzdialenými úsekmi sekvencie bolo nutné vytvoriť klasifikátor, ktorý ohodnotí možnú interakciu medzi danými dvoma aminokyselinami v sekvencii. Z tohto dôvodu sme na príkladoch úsekov s interagujúcimi aminokyselinami natrénovali viacero klasifikátorov pomocou metódy support vector machines. Vytvorili sme viacero SVM modelov s rôznymi parametrami a z nich sme na základe porovnania ich klasifikačnej sily vybrali modely, ktoré sme ďalej použili pre skórovanie zarovnania deskriptoru k sekvencii.

Po určení skórovacej schémy pre zarovnanie deskriptora k sekvencii sme riešili problém nájdenia najlepšieho takéhoto zarovnania pre daný deskriptor a sekvenciu. Na riešenie problému zarovnania sme najskôr navrhli celočíselný lineárny program. Tento program sme otestovali na skupine proteínov, ktorá obsahovala proteíny s doménou Telo_bind, ako aj proteíny bez nej. Zarovnania proteínov, ktoré obsahovali doménu popísanú deskriptorom, dosahovali výrazne vyššie skóre, ako proteíny bez hľadanej domény, takže táto metóda dokázala identifikovať prítomnosť hľadanej domény v proteínoch. Avšak ohodnotenie negatívnych príkladov bolo príliš časovo náročné, výpočet pre veľkú časť z týchto príkladov trval niekoľko dní. Použitie tejto metódy je teda nepraktické v prípadoch, kedy je potrebné zarovnať deskriptor k veľkému množstvu sekvencií.

Keďže deskriptor môže popisovať ľubovoľné interakcie medzi vzdialenými úsekmi sekvencie, je tento problém analogický k problému zarovnania RNA deskriptora, ktorý je NP-ťažký. Z tohto dôvodu riešenie všeobecného problému zarovnania môže byť časovo príliš náročné. Problém zarovnania sme zjednodušili zavedením obmedzenia na prípustné väzby v deskriptore. Tento zjednodušený problém sme riešili pomocou dynamického programovania, ktoré dokázalo nájsť zarovnanie deskriptora aj k dlhším sekvenciám v prijateľnom čase. Vďaka tomuto zjednodušeniu je možné aplikovať hľadanie deskriptorom aj na úrovni celého genómu.

Pomocou tohto algoritmu sme otestovali deskritor pre Telo_bind doménu na sade 180 proteínov a porovnali sme naše výsledky s programom Hmmer, ktorý používa na reprezentáciu domény profilové HMM. Metóda hľadania domén pomocou deskriptora dokáže korektne rozlíšiť proteíny s doménou Telo_bind od proteínov bez tejto domény. Navyše výhodou našej metódy je jej schopnosť identifikovať aj proteíny s doménami, ktoré sú príbuzné k hľadanej doméne Telo_bind. Túto metódu je možné použiť na analýzu celých genómov, čo sme demonštrovali na genóme kvasinky *Yarrowia lipolytica*.

Metóda, ktorú sme predstavili v tejto práci, môže byť základom pre ďalší výskum, ktorý môže viesť viacerými smermi. V našich testoch sa viackrát objavil výsledok s veľmi dobrým skóre, ktoré bolo výsledkom zarovnania coil segmentov v deskriptore k dlhým coil úsekom v sekvencii. Keďže popisovaná doména je charakterizovaná najmä zoskupením β-listov do β-barelu, tieto výsledky sú pravdepodobne falošne pozitívne. Jedným z cieľov ďalšieho výskumu môže byť vylepšenie popísanej metódy o rôzne filtre na dodatočné spracovanie výsledkov, ktoré by mohli odhaliť takéto falošne pozitívne výsledky. Iným spôsobom riešenia tohto problému môže byť úprava skórovania zarovnania, či už normalizáciou skóre segmentu podľa jeho dĺžky, alebo ováhovanie skóre pre sekundárnu štruktúru pre jednotlivé segmenty v deskriptore.

V našej práci sme na optimalizáciu časovej náročnosti použili orezávanie výpočtu pre proteíny, ktoré nemajú β-listy na umiestnenie hydrogénových väzieb predikované s dostatočne veľkou pradepodobnosťou. Toto orezávanie nám umožňuje zrýchliť výpočet zarovnania a tak nám dovoľuje použiť túto metódu na veľké množstvo kandidátskych sekvencií. Na druhej strane, keďže predikcia sekundárnej štruktúry nemusí správne klasifikovať všetky úseky s β-listami vo vstupnej sekvencii, podmienka, ktorú používame na orezanie výpočtu, môže odstrániť z výsledkov aj pozitívne výskyty domény. Prah, pri ktorom sú medzivýsledky v zarovnaní orezané, bol určený ad hoc a ďalšia práca by sa mohla zamerať na určenie tohto prahu systematickým spôsobom z dostupných pozitívnych príkladov hľadanej domény. Alternatívny prístup k tomuto problému môže byť aj zrýchlenie klasifikácie potenciálnych hydrogénových väzieb, ktoré by umožnilo použitie našej metódy na veľkom počte sekvencií bez potreby orezávania výpočtu.

Váhy jednotlivých zložiek skóre, ktoré sme použili v našich testoch, boli určené ručne na základe výsledkov pre testovaciu sadu 23 proteínov. Rozšírením našej metódy by mohlo byť iteratívne určenie týchto váh pomocou danej trénovacej množiny. Cieľom by bolo nájsť váhy, ktoré budú maximalizovať rozdiel medzi dosiahnutým skóre pozitívnych a negatívnych príkladov. Takto získané váhy by mohli pomôcť zlepšiť citlivosť hľadania.

Dynamický program, ktorý sme v práci popísali, rieši relaxovaný problém len pre triedu deskriptorov spĺňajúcich obmedzenie na hydrogénové väzby. V našej práci sme použili deskriptor pre Telo_bind doménu, v ktorom sme museli vynechať jednu väzbu, aby spĺňal toto obmedzenie. Vynechanú väzbu sme na záver dopočítali do zarovnania a neočakávame, že by jej vynechaním pri určení zarovnania došlo k výraznej strate informácií charakterizujúcich hľadanú doménu. V prípade iných domén však toto obmedzenie na štruktúru väzieb, ktoré dokážeme popísať v deskriptore, môže byť príliš limitujúce, takže deskriptory reprezentujúce tieto domény nedokážu vystihnúť vzťahy medzi vzdialenými aminokyselinami v doméne. Jedným z možných teoretických problémov je lepšia charakterizácia tried deskriptorv, ktoré možno zarovnávať ku sekvenciám v polynomiálnom čase, ako aj tvorba algoritmov, ktoré takéto zarovnania počítajú. Vytvorené zarovnania by bolo potom možné použiť na praktické zarovnávanie bohatšej triedy deskriptorov.

Ďalšia práca by sa tiež mohla sústrediť na rozšírenie použitia popísanej metódy pri analýze biologických dát. V tejto práci sme vytvorili a používali deskriptor pre Telo_bind doménu z rodiny OB-fold domén. Pre ďalší výskum by bolo vhodné vytvoriť deskriptory pre rôzne ďalšie domény so špecifickou štruktúrou (napríklad β-propeler), ktoré by mohli priniesť zaujímavé výsledky pri hľadaní evolučne vzdialenejších ortológov. Dlhodobým cieľom by mohlo byť vytvorenie databázy s deskriptormi pre domény s charakteristickou štruktúrou a ďalších algoritmov, ktoré by využívali tieto deskriptory pre hľadanie ortológov alebo anotáciu funkcie proteínov v novoosekvenovaných organizmoch.

Literatúra

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

Arcus, V. (2002). Ob-fold domains: a snapshot of the evolution of sequence, structure and function. *Current opinion in structural biology*, 12(6):794–801.

Bailey, T. and Elkan, C. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proc Int Conf Intell Syst Mol Biol*, volume 2, pages 28–36. Citeseer.

Bailey, T. and Gribskov, M. (1998). Combining evidence using p-values: application to sequence homology searches. *Bioinformatics*, 14(1):48.

Bailey, T., Williams, N., Misleh, C., and Li, W. (2006). MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic acids research*, 34(Web Server issue):W369.

Bairoch, A., Boeckmann, B., Ferro, S., and Gasteiger, E. (2004). Swiss-prot: juggling between evolution and stability. *Briefings in Bioinformatics*, 5(1):39–55.

Blackburn, E., Greider, C., and Szostak, J. (2006). Telomeres and telomerase: the path from maize, Tetrahymena and yeast to human cancer and aging. *Nature medicine*, 12(10):1133–1138.

Botstein, D., Chervitz, S., and Cherry, J. (1997). Yeast as a model organism. *Science-AAAS-Weekly Paper Edition*, 277(5330):1259–1259.

Cameron, M., Williams, H., and Cannane, A. (2004). Improved gapped alignment in BLAST. *IEEE Transactions on Computational Biology and Bioinformatics*, pages 116–129.

Chandra, A., Hughes, T., Nugent, C., and Lundblad, V. (2001). Cdc13 both positively and negatively regulates telomere replication. *Genes & Development*, 15(4):404.

Chang, C. (2005). BLAST implementation on BEE2. *Electrical Engineering and Computer Science University of California at Berkeley*.

Chiang, D., Joshi, A., and Searls, D. (2006). Grammatical representations of macromolecular structure. *Journal of Computational Biology*, 13(5):1077–1100.

Cplex (2010). Ibm ilog cplex interactive optimizer 12.2.0.0. www-01.ibm.com/software/integration/optimization/cplex-optimizer.

Dantzig, G. (1951). Maximization of a linear function of variables subject to linear inequalities. *New York*.

Dayhoff, M., Schwartz, R., and Orcutt, B. (1978). A model of evolutionary change in proteins. *Atlas of protein sequence and structure*, 5(Suppl 3):345–352.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis*. Cambridge Univ. Press.

Eddy, S. (1998). Profile hidden Markov models. *Bioinformatics*, 14(9):755.

Eddy, S. R. (2011). Accelerated Profile HMM Searches. *PLoS Comput Biol*, 7(10):e1002195.

Edgar, R. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792.

Finn, R., Tate, J., Mistry, J., Coggill, P., Sammut, S., Hotz, H., Ceric, G., Forslund, K., Eddy, S., Sonnhammer, E., et al. (2007). The Pfam protein families database. *Nucleic acids research*.

Fitch, W. (2000). Homology:: a personal view on some of the problems. *Trends in genetics*, 16(5):227–231.

Francino, M. (2005). An adaptive radiation model for the origin of new gene functions. *Nature Genetics*, 37(6):573–578.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer Series in Statistics.

Frith, M., Saunders, N., Kobe, B., and Bailey, T. (2008). Discovering sequence motifs with arbitrary insertions and deletions. *PLoS Computational Biology*, 4(5).

Gabaldón, T., Rainey, D., and Huynen, M. (2005). Tracing the evolution of a large protein complex in the eukaryotes, NADH: ubiquinone oxidoreductase (Complex I). *Journal of molecular biology*, 348(4):857–870.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705.

Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. U.S.A.*, 84:4355–4358.

Haussler, D., Krogh, A., Mian, I., and Sjolander, K. (1993). Protein modeling using hidden Markov models: Analysis of globins. In *System Sciences, 1993, Proceeding of the Twenty-Sixth Hawaii International Conference on*, volume 1.

Hayashi, N. and Murakami, S. (2002). STM1, a gene which encodes a guanine quadruplex binding protein, interacts with CDC13 in Saccharomyces cerevisiae. *Molecular Genetics and Genomics*, 267(6):806–813.

Henikoff, S. and Henikoff, J. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915.

Hirsh, A. and Fraser, H. (2001). Protein dispensability and rate of evolution. *Nature*, 411(6841):1046–1048.

Hughes, T., Evans, S., Weilbaecher, R., and Lundblad, V. (2000). The Est3 protein is a subunit of yeast telomerase. *Current Biology*, 10(13):809–812.

Jimenez, R., Rampášek, L., Brejová, B., Vinař, T., and Lupták, A. (2012). Discovery of rna motifs using a computational pipeline that allows insertions in paired regions and filtering of candidate sequences. *Methods in molecular biology (Clifton, NJ)*, 848:145.

Jmol (2012). Jmol: an open-source Java viewer for chemical structures in 3D. www.jmol.org.

Joachims, T. (1999). *Making large-Scale SVM Learning Practical*. MIT-Press.

Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol*, 292(2):195–202.

Kanehisa, M. (2002). The KEGG database. *Silico Simul Biol Proc*, 247:91–103.

Kellis, M., Patterson, N., Birren, B., Berger, B., and Lander, E. (2004). Methods in comparative genomics: genome correspondence, gene identification and regulatory motif discovery. *Journal of Computational Biology*, 11(2-3):319–355.

Koski, L. and Golding, G. (2001). The closest BLAST hit is often not the nearest neighbor. *Journal of Molecular Evolution*, 52(6):540–542.

Krogh, A., Brown, M., Mian, I., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology. Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531.

Larkin, M., Blackshields, G., Brown, N., Chenna, R., McGettigan, P., McWilliam, H., Valentin, F., Wallace, I., Wilm, A., Lopez, R., et al. (2007). Clustal W and Clustal X version 2.0. *Bioinformatics*, 23(21):2947.

Lathrop, R. (1994). The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein engineering*, 7(9):1059–1068.

Lee, J., Mandell, E., Tucey, T., Morris, D., and Victoria, L. (2008). The Est3 protein associates with yeast telomerase through an OB-fold domain. *Nature structural & molecular biology*, 15(9):990.

Lendvay, T., Morris, D., Sah, J., Balasubramanian, B., and Lundblad, V. (1996). Senescence mutants of Saccharomyces cerevisiae with a defect in telomere replication identify three additional EST genes. *Genetics*, 144(4):1399.

Leslie, C., Eskin, E., and Noble, W. (2002). The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, pages 566–575. Hawaii, USA.

Li, L., Stoeckert, C., and Roos, D. (2003). OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome research*, 13(9):2178.

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659.

Linger, B. and Price, C. (2009). Conservation of telomere protein complexes: shuffling through evolution. *Critical Reviews in Biochemistry and Molecular Biology*, (00):1–13.

Lingner, J., Cech, T., Hughes, T., and Lundblad, V. (1997). Three Ever Shorter Telomere (EST) genes are dispensable for in vitro yeast telomerase activity. *Proceedings of the National Academy of Sciences*, 94(21):11190.

Lodish, H. (2003). *Molecular cell biology*. WH Freeman.

Lustig, A. (2001). Cdc13 subcomplexes regulate multiple telomere functions. *Nature structural biology*, 8(4):297.

Mamitsuka, H. and Abe, N. (1994). Predicting location and structure of beta-sheet regions using stochastic tree grammars. *ISMB-94*, pages 276–284.

Maxam, A. and Gilbert, W. (1977). A new method for sequencing dna. *Proceedings of the National Academy of Sciences*, 74(2):560.

Meier, B., Driller, L., Jaklin, S., and Feldmann, H. (2001). New function of CDC13 in positive telomere length regulation. *Molecular and Cellular Biology*, 21(13):4233.

Menke, M., Berger, B., and Cowen, L. (2010). Markov random fields reveal an n-terminal double beta-propeller motif as part of a bacterial hybrid two-component sensor system. *Proceedings of the National Academy of Sciences*, 107(9):4069–4074.

Moreno-Hagelsieb, G. and Latimer, K. (2008). Choosing BLAST options for better detection of orthologs as reciprocal best hits. *Bioinformatics*, 24(3):319.

Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Nosek, J. and Tomáška, L. (2008). *Origin and evolution of telomeres*. Landes Bioscience.

Ohno, S. et al. (1970). *Evolution by gene duplication*. London: George Alien & Unwin Ltd. Berlin, Heidelberg and New York: Springer-Verlag.

Olmea, O., Rost, B., and Valencia, A. (1999). Effective use of sequence correlation and conservation in fold recognition1. *Journal of molecular biology*, 293(5):1221–1239.

Qi, H. and Zakian, V. (2000). The Saccharomyces telomere-binding protein Cdc13p interacts with both the catalytic subunit of DNA polymerase α and the telomerase-associated Est1 protein. *Genes & Development*, 14(14):1777.

Rampasek, L. (2011). RNA structural motif search is NP-complete. In *Studentska vedecka konferencia FMFI UK, Bratislava*, pages 341–348.

Remm, M., Storm, C., and Sonnhammer, E. (2001). Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of molecular biology*, 314(5):1041–1052.

Rivera, M., Jain, R., Moore, J., and Lake, J. (1998). Genomic evidence for two functionally distinct gene classes. *Proceedings of the National Academy of Sciences*, 95(11):6239.

Rose, P. W., Beran, B., Bi, C., Bluhm, W. F., Dimitropoulos, D., Goodsell, D. S., Prlic, A., Quesada, M., Quinn, G. B., Westbrook, J. D., Young, J., Yukich, B., Zardecki, C., Berman, H. M., and Bourne, P. E. (2011). The RCSB Protein Data Bank: redesigned web site and web services. *Nucleic Acids Res*, 39(Database issue):D392–401.

Rost, B., Sander, C., and Schneider, R. (1994). Phd-an automatic mail server for protein secondary structure prediction. *Computer applications in the biosciences: CABIOS*, 10(1):53–60.

Sanger, F. and Coulson, A. (1975). A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *Journal of molecular biology*, 94(3):441–448.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons Inc.

Shachar, R., Ungar, L., Kupiec, M., Ruppin, E., and Sharan, R. (2008). A systems-level approach to mapping the telomere length maintenance gene circuitry. *Molecular Systems Biology*, 4:172.

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

Steward, R. and Thornton, J. (2002). Prediction of strand pairing in antiparallel and parallel β-sheets using information theory. *Proteins: Structure, Function, and Bioinformatics*, 48(2):178–191.

Tatusov, R., Galperin, M., Natale, D., and Koonin, E. (2000). The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28(1):33.

Taylor, J. and Raes, J. (2004). Duplication and divergence: the evolution of new genes and old ideas.

Theobald, D., Mitton-Fry, R., and Wuttke, D. (2003). Nucleic acid recognition by ob-fold proteins. *Annual review of biophysics and biomolecular structure*, 32:115.

Tomáška, L. (2009). Elektronická korešpondencia.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Waldispühl, J., Berger, B., Clote, P., and Steyaert, J. (2006). Predicting transmembrane β-barrels and interstrand residue interactions from sequence. *PROTEINS: Structure, Function, and Bioinformatics*, 65(1):61–74.

Wall, D., Fraser, H., and Hirsh, A. (2003). Detecting putative orthologs. *Bioinformatics*, 19(13):1710.

Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348.

Wapinski, I., Pfeffer, A., Friedman, N., and Regev, A. (2007). Automatic genome-wide reconstruction of phylogenetic gene trees. *Bioinformatics*, 23(13):i549.

Waterman, M. and Eggert, M. (1987). A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *Journal of Molecular Biology*, 197(4):723.

Wilbur, W. and Lipman, D. (1983). Rapid similarity searches of nucleic acid and protein data banks. *Proceedings of the National Academy of Sciences*, 80(3):726.

Wolsey, L. (1980). Heuristic analysis, linear programming and branch and bound. *Combinatorial Optimization II*, pages 121–134.

Yang, Z. (2000). Phylogenetic analysis by maximum likelihood (PAML). *University College, London*.

Zimmermann, K. (2003). Tertiary structure prediction. *An Introduction to Protein Informatics*, pages 169–216.