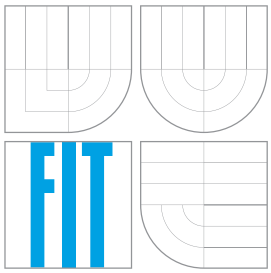
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VYHLEDÁVÁNÍ PŘÍBUZNÝCH PROTEINŮ S MODIFIKOVANOU FUNKCÍ

DETECTION OF RELATED PROTEINS WITH MODIFIED FUNCTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ HON

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ MARTÍNEK, Ph.D.

BRNO 2015

Abstrakt

Proteinové inženýrství je dynamicky se rozvíjecí obor s velkým množstvím potenciálních aplikací v praxi. Úspěch v tomto oboru je však podmíněn co nejlepším využitím všech dostupných informací o proteinech, k čemuž se využívá množství bioinformatických nástrojů a analýz. Cílem této práce je vytvoření nového nástroje na podporu proteinového inženýrství, který by umožnil vyhledávání příbuzných proteinů s modifikovanou funkcí ve stále rostoucích proteinových databázích. Návrh tohoto nástroje je koncipován jako spojení řady existujících bioinformatických analýz a umožní identifikovat příbuzné proteiny se stejným typem enzymatické funkce, avšak s mírně modifikovanými vlastnostmi, především z hlediska selektivity, reakční rychlosti a stability.

Abstract

Protein engineering is a young dynamic discipline with great amount of potential practical applications. However, its success is primarily based on perfect knowledge and usage of all existing information about protein function and structure. To achieve that, protein engineering is supported by plenty of bioinformatic tools and analysis. The goal of this project is to create a new tool for protein engineering that would enable researchers to identificate related proteins with modified function in still growing biological databases. The tool is designed as an automated workflow of existing bioinformatic analyses that leads to identification of proteins with the same type of enzymatic function, but with slightly modified properties – primarily in terms of selectivity, reaction speed and stability.

Klíčová slova

vyhledávání příbuzných proteinů, predikce funkce proteinu, bioinformatika

Keywords

related protein identification, protein function prediction, bioinformatics

Citace

Jiří Hon: Vyhledávání příbuzných proteinů
s modifikovanou funkcí, diplomová práce, Brno, FIT VUT v Brně, 2015

Vyhledávání příbuzných proteinů s modifikovanou funkcí

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod odborným vedením pana Ing. Tomáše Martínka Ph.D.

.....
Jiří Hon
23. května 2015

Poděkování

Děkuji za pomoc a odborný přístup vedoucímu práce, panu Ing. Tomáši Martínkovi, Ph.D. Dále děkuji celému týmu Loschmidtových laboratoří a obzvláště paní Mgr. Evě Šebestové, Ph.D. za náměty týkající se navrženého nástroje. Velice si vážím času, který mi věnovali při práci na tomto diplomovém projektu.

Výpočetní prostředky byly poskytnuty MetaCentrem v rámci programu LM2010005 a organizací CERIT-SC v rámci programu Centre CERIT Scientific Cloud, který je součástí Operačního programu výzkum a vývoj pro inovace, reg. č. CZ.1.05/3.2.00/08.0144.

Obsah

Kapitola 1

Úvod

Už od třicátých let 19. století, kdy proteiny poprvé pojmenoval významný švédský chemik Jöns Jacob Berzelius, jsou předmětem bádání mnoha vědců na celém světě. Proteiny jsou z chemického hlediska nejsložitější a funkčně nejdůmyslnější známé organické molekuly a jsou považovány za podstatu života na zemi[2]. Proto, chceme-li dobře porozumět procesům, které se odehrávají v tělech živých organismů, musíme nejdříve zodpovědět mnoho složitých otázek: Jakou mají tyto molekuly strukturu a jak se jejich struktura utváří? Jakým způsobem je realizována jejich funkce? Jaké jsou mezi nimi vztahy a interakce?

Pokud by se podařilo dokonale zodpovědět všechny otázky týkající se proteinů, mohli bychom využít těchto znalostí k vývoji nových proteinů, u nichž si můžeme představit celou řadu aplikací – od léčby dosud neléčitelných nemocí, přes ochranu životního prostředí (např. formou katalýzy rozkladu toxických látek), po produkci biopolymerů a nanobiotechnologie. Vědní disciplína, která se těmito aplikacemi zabývá, se nazývá proteinové inženýrství a snaží se co nejlépe aplikovat všechny dosud známé částečné odpovědi na výše uvedené otázky a využít je k vývoji proteinů s modifikovanými vlastnostmi.

Metody používané v proteinovém inženýrství se nespoléhají pouze na znalosti odborníků v oboru, ale i na celou řadu bioinformatických analýz, které mohou o proteinech a jejich funkci napovědět množství hodnotných informací. Existují například výpočetní nástroje pro vícenásobné zarovnání proteinových sekvencí, které mohou být použity pro analýzu stupně evoluční konzervovanosti jednotlivých aminokyselin, dále nástroje pro predikci struktury nebo výpočet strukturního zarovnání proteinů (viz kapitola 4).

Cílem tohoto diplomového projektu je pak navrhnout nový nástroj na podporu proteinového inženýrství, který by umožňoval vyhledávat v databázích biologických sekvencí příbuzné proteiny s modifikovanou funkcí, přičemž modifikovanou funkcí v tomto případě není myšlen odlišný typ funkcionality proteinu (např. změna katalyzovaného substrátu u enzymů), ale její modifikace ve smyslu rychlosti reakce, stability proteinu apod.

Smysl takového nástroje je založen na hypotéze, že počet proteinových sekvencí v biologických databázích nadále roste vysokým tempem (měsíčně zhruba milión nových sekvencí) a bylo by proto užitečné vytvořit takový nástroj, který by v nich dokázal hledat nové zajímavé sekvence, které bychom např. mohli zahrnout jako nové vstupy pro metody proteinového inženýrství a tím zvýšit šanci na úspěch při vývoji proteinu s vylepšenou funkcí.

Tato technická zpráva je organizována do několika kapitol. V kapitole 2 jsou představeny základní informace o proteinech nutné pro orientaci v dalším textu. Následuje kapitola 3 uvádějící přehled metod používaných v proteinovém inženýrství. V kapitole 4 jsou popsány existující nástroje využitelné pro hledání příbuzných proteinů s modifikovanou funkcí, v ka-

pitole 5 je pak uveden stručný návrh nového nástroje k řešení tohoto problému, který je podrobněji rozveden v kapitole 6 věnující se implementaci. Poslední kapitola 7 popisuje ověření funkce implementovaného nástroje na konkrétní rodině enzymů haloalkan dehalogenáz.

Kapitola 2

Proteiny a jejich funkce

V této kapitole jsou uvedeny některé esenciální informace o proteinech, jejichž znalost se předpokládá dále ve zbývajících kapitolách této práce.

2.1 Struktura proteinu

Molekula proteinu je tvořena řetězcem aminokyselin spojených se svými sousedy kovalentní peptidovou vazbou. Na každé pozici tohoto řetězce můžeme nalézt jednu z 21 možných aminokyselin. Prostorovou strukturu proteinů můžeme analyzovat na několika úrovních podobně jako je tomu u molekul RNA nebo DNA.

Pořadí aminokyselin v tomto řetězci nazýváme *primární* strukturou proteinu. Na vyšší úrovni je možné mluvit o tzv. *sekundární* struktuře, čímž rozumíme lokální prostorové uspořádání polypeptidového řetězce mezi několika po sobě jdoucími aminokyselinami. Rozlišujeme několik základních motivů sekundární struktury: alfa šroubovici (alfa-helix) připomínající kanonickou strukturu DNA, strukturu skládaného listu (beta-sheet) a smyčky (coil). *Terciární* struktura proteinu pak definuje trojrozměrné uspořádání *celého* proteinu v prostoru.

2.2 Funkce proteinu

Proteiny plní v živých organizmech velké množství rozmanitých funkcí. Pro ilustraci uvádíme základní přehled některých obecných funkcí:[2]

- Strukturní – mechanická podpora buňkám a tkáním.
- Transportní – přenos malých molekul a iontů.
- Pohybová – pohyb buněk a tkání.
- Zásobní – skladování malých molekul nebo iontů.
- Signální – přenos informačních signálů z buňky do buňky.
- Receptorová – detekce fyzikálních a chemických signálů a předávání ke zpracování buňce.
- Regulační – vazba na DNA a spouštění resp. vypínání procesu transkripce.

- Enzymatická – katalýza rozpadu a tvorby kovalentních vazeb.

Z pohledu této práce se dále budeme zabývat především proteiny s enzymatickou funkcí (dále jen enzymy).

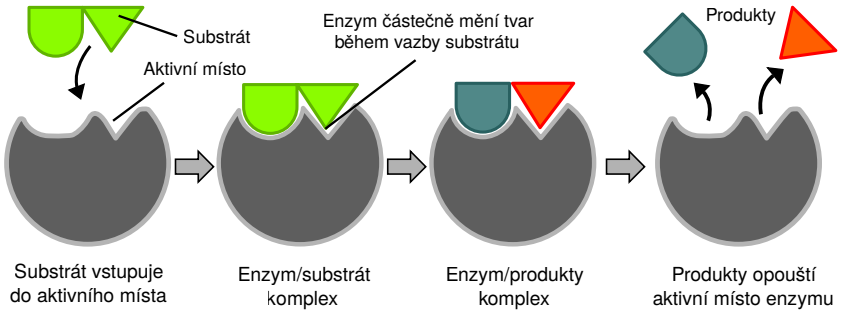
2.3 Enzymy

Živé buňky obsahují tisíce různých enzymů, z nichž každý katalyzuje specifickou reakci. Například enzym pepsin odbourává v žaludku proteiny přijímané v potravě nebo DNA-polymeráza syntetizuje DNA[2]. Pro všechny enzymy je společné, že svým způsobem umožňují, aby reakce za daných podmínek proběhla. Bez jejich přítomnosti by např. bylo nutné do reakce přidat větší množství tepla nebo úplně změnit jiné reakční podmínky (hodnotu pH, tlak, apod.).

Enzymy nemusí být vždy jen aminokyselinové řetězce (proteiny). Může se jednat i o molekuly RNA nebo o komplexy aminokyselinových řetězců a molekul RNA. Na druhou stranu však většinu enzymů tvoří právě proteiny, resp. komplexy proteinů.

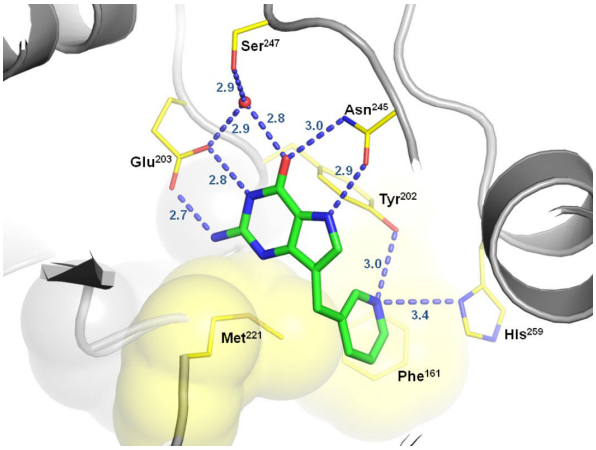
Model klíče a zámku

Enzymy jsou obvykle vysoce specifické vzhledem k reakcím, které katalyzují a vzhledem k substrátům, které do těchto reakcí vstupují. Tato specificita je dána komplementárním tvarem, nábojem a hydrofilní resp. hydrofobní charakteristikou enzymů a substrátů[18]. Tento fenomén se obvykle vysvětluje na modelu klíče a zámku, kde zámek představuje *aktivní místo* enzymu a klíčem je *kompatibilní substrát* (viz obrázek 2.1). Ačkoliv tento model jednoduchým způsobem vysvětluje substrátovou specificitu, selhává při vysvětlení přechodného stavu (konformace), do kterého se enzym obvykle transformuje během reakce, aby umožnil její správný průběh.



Obrázek 2.1: *Upravený model klíče a zámku.*[56] Tvar aktivního místa je částečně indukován tvarem substrátu – v literatuře se tento jev někdy vysvětluje na modelu ruky a rukavice, na němž si lze lépe představit průběh a limity částečné indukce.

V šedesátých letech byla navržena mírná modifikace tohoto modelu. Jelikož jsou enzymy spíše pružné molekuly, tvar aktivního místa (zámku) se během vazby substrátu (klíče) průběžně mění[26]. Proto se substrát neváže jednoduše na pevně definované aktivní místo, ale postranní řetězce aminokyselin, které tvoří aktivní místo, se postupně formují do tvaru, který umožní katalytickou funkci daného enzymu. V některých případech se do jisté míry mění i tvar substrátu.



Obrázek 2.2: Prostorová ilustrace aktivního místa enzymu *Sm*PNP[40]. Aktivní místo je oblast na povrchu nebo uvnitř enzymu, na kterou se váže substrát a probíhá v něm katalyzovaná reakce. Žlutou barvou jsou označena tzv. katalytická rezidua, tj. aminokyseliny, bez jejichž přítomnosti a správné pozice nemůže reakce proběhnout. Zelená molekula značí navázaný substrát – v tomto konkrétním případě se jedná o inhibitor aktivního místa.

Aminokyseliny v aktivním místě, které se účastní katalýzy substrátu, nazýváme *katalytická rezidua* (viz obrázek 2.2). Jejich znalost je kritická zejména pro pochopení funkce daného enzymu a v neposlední řadě pro návrh modifikací, které by mohly vést k lepším vlastnostem studovaného enzymu (viz kapitola 3).

Mechanismus

Enzymy mohou katalyzovat reakce různými mechanismy. Všechny nicméně spojuje, že snižují tzv. Gibbsovu volnou energii katalyzované reakce[16]. Následuje několik příkladů takových mechanismů:

- Snížení aktivační energie vytvořením takového prostředí, ve kterém je stabilní přechodný stav substrátu – např. deformováním navázaného substrátu do tvaru, ve kterém je zapotřebí menší energie k provedení reakce.
- Vytvoření alternativní reakční cesty – rozdělení reakce do více fází s meziprodukty.
- Snížení reakční entropie přiblížením substrátů a zajištěním jejich správné orientace, aby spolu mohly reagovat.
- Zrychlení reakce za daných teplotních podmínek – funguje do určité maximální teploty, při které začne docházet k *denaturaci* proteinu, neboli nevratné změně tvaru proteinu.

Kapitola 3

Proteinové inženýrství

Proteinové inženýrství je vědní disciplína, která se zabývá procesem vývoje nových typů proteinů. Jelikož tento diplomový projekt předpokládá využití výsledku právě v této oblasti, je pro orientaci uveden přehled aktuálních metod využívaných v proteinovém inženýrství spolu se souhrnem vybraných aplikací.

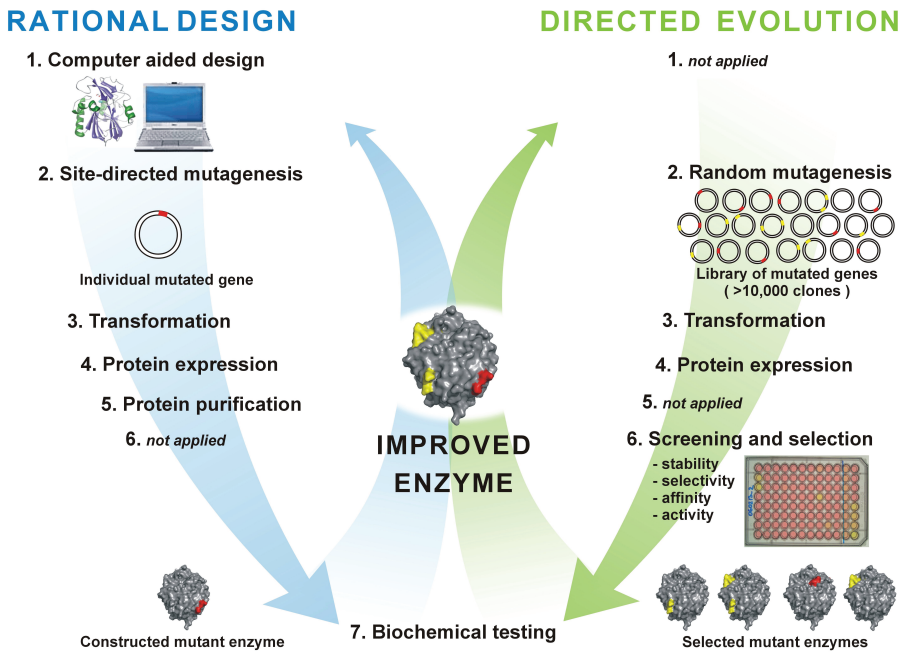
Obecně vzato existují dva přístupy k vývoji proteinů s vylepšenými resp. modifikovanými vlastnostmi: racionální design a řízená evoluce[55]. Tyto techniky se nutně vzájemně nevylučují, naopak je vhodné oba přístupy kombinovat.

Racionální design vychází z detailní znalosti struktury a funkce modifikovaného proteinu, na základě kterých je možné navrhnout konkrétní změny v proteinu. Například záměnu některých aminokyselin pomocí techniky místně specifické mutageneze. Na jednu stranu je laboratorní provedení této metody poměrně snadné, na druhou stranu ale obvykle nejsou dostupné kvalitní strukturní informace o studovaném proteinu. Kvalita výsledků získaných metodou racionálního designu stojí a padá na kvalitě analýz a predikcí nad známou strukturou proteinu. K tomuto slouží celá řada bioinformatických nástrojů a algoritmů, zejména nástroje pro modelování a predikci volné energie modifikovaného proteinu, nástroje pro dokování molekul substrátu do aktivního místa proteinu, algoritmy pro predikci škodlivosti zamýšlených aminokyselinových mutací, apod.

Oproti tomu metoda *řízené evoluce* ve své původní podobě nevyužívá žádnou z počítačových analýz. Místo toho se snaží zrychleně simulovat průběh evoluce (od toho název řízená evoluce) metodou náhodné mutageneze. Jedná se o proces, při němž je vytvořeno velké množství mutantů s náhodně mutovanými aminokyselinami, z nichž se pak vybírají jedinci s nejlepšími požadovanými vlastnostmi. Je zřejmé, že tato metoda je velice náročná na laboratorní práci a zpravidla vyžaduje robotické vybavení pro testování tak velkého množství variant. Na druhou stranu je tímto způsobem možné vylepšovat protein, o němž nejsou k dispozici strukturní informace.

Společným rysem obou metod je jejich iterativní průběh (viz obrázek 3.1). Pokud se v dané iteraci podaří nalézt protein s lepšími vlastnostmi, může být vhodné jej použít jako výchozí protein pro další iteraci.

Současné metody v proteinovém inženýrství využívají výhod obou přístupů[31]. Pokud totiž máme aspoň částečné informace o struktuře nebo funkci proteinu, mohou nám posloužit k tomu, abychom snížili množství potenciálních reziduí, které je vhodné mutovat. Například budeme řízenou evolucí mutovat jen ta rezidua, která přicházejí do kontaktu se substrátem v aktivním místě. Výsledný počet možných mutantů (variant) proteinu se výrazně sníží a tím i množství nutné laboratorní práce. Tyto metody obvykle nazýváme semi-racionální, protože stojí na pomezí racionálního designu a řízené evoluce.



Obrázek 3.1: *Srovnání racionálního designu a řízené evoluce.*[41]

3.1 Aplikace

Proteinové inženýrství je velmi progresivní obor a otevírá prostor pro velké množství nových užitečných aplikací. Pro lepší představu jsou uvedeny některé z významných aplikací v různých oborech lidských činností spolu s odkazy do literatury:[54]

- Potravinářský průmysl[23, 25, 1]
- Průmysl věnující se detergentům[21]
- Ochrana životního prostředí[11, 4]
- Lékařství[59, 38]
- Produkce biopolymerů[44, 6]
- Nanobiotechnologie[46, 53]

3.2 Nový nástroj na podporu racionálního designu

Obvykle se k racionálnímu designu přistupuje způsobem, že je vybrán nejvhodnější známý protein a ten se iterativně vylepšuje. Problém nastává ve chvíli, kdy se ukáže, že zvolený protein nebyl k proteinovému inženýrství vhodný a snahy o modifikaci funkce nevedou k lepším výsledkům. Vyvstává otázka, jak vybrat jiný příbuzný protein, pravděpodobně s modifikovanou funkcí, se kterým by bylo možné dosáhnout lepších výsledků. Za předpokladu, že databáze proteinových sekvencí rostou vysokým tempem (donedávna exponenciálně),

je možné se domnívat, že se v nich v budoucnu objeví nový příbuzný protein, například z jiného organismu, který by byl vhodnější než aktuálně studovaný protein.

Vyvstává tedy potřeba bioinformatického nástroje, který bude schopen v proteinových databázích hledat příbuzné proteiny s modifikovanou funkcí, přičemž modifikovanou funkcí v tomto případě není myšlen odlišný typ funkcionality proteinu (např. úplná změna katalyzovaného substrátu u enzymů), ale pouze její modifikace ve smyslu rychlosti reakce, stability proteinu apod. Návrh a implementace takového nástroje je proto cílem tohoto diplomového projektu.

Kapitola 4

Existující přístupy

Úvodem této kapitoly je vhodné uvést, že se při nejlepší vůli nepodařilo v současné literatuře najít zmínku o žádném již existujícím nástroji pro vyhledávání příbuzných proteinů s modifikovanou funkcí. V některých článcích sice lze objevit analýzy, kde se autoři snažili vyhledat příbuzné proteiny a ověřit jejich funkci[47], ale nejednalo se nikdy o obecný postup nebo algoritmus, který by byl použitelný i pro jiné typy proteinů než ty, na něž byla konkrétní studie zaměřena.

Zdá se tedy, že se zatím nikdo na problém nepodíval z nového úhlu pohledu. Když si totiž zadání problému rozdělíme na dvě části, za prvé na problém hledání příbuzných proteinů a za druhé na problém predikce funkce proteinu, pak v každé z těchto oblastí existuje dostatečné množství literatury a nástrojů. Zatím pouze nikdo nevytvořil nástroj, který by propojoval existující přístupy pro vyhledávání příbuzných proteinů s predikcí jejich funkce, aby ve výsledku vedly k vyhledávání příbuzných proteinů s modifikovanou funkcí.

V této kapitole jsou proto představeny některé z existujících přístupů k vyhledávání příbuzných proteinů na základě sekvence a také metody pro predikci funkce proteinu, aby pak v následující kapitole byl navržen nástroj, který bude obě oblasti propojovat v automatizovanou analýzu.

4.1 Nástroje pro hledání homologních sekvencí

Představeny jsou nejpoužívanější nástroje pro hledání homologních sekvencí počínaje nástrojem BLAST, přes PatternHunter až k nejmladšímu nástroji HHblits.

BLAST

Nástroj BLAST (Basic Local Alignment Search Tool)[3] je jedním z aktuálně nejpoužívanějších nástrojů pro vyhledávání příbuzných sekvencí v biologických databázích, a proto si jeho princip popíšeme podrobněji. Umožňuje pracovat jak se sekvencemi aminokyselin, tak i se sekvencemi nukleotidů. Jeho vyhledávací algoritmus aproximuje funkci algoritmu Smith-Waterman[51] a díky použité heuristice je mnohonásobně rychlejší při zachování rozumné přesnosti. Algoritmus funguje v několika krocích:

1. V první fázi se odstraní oblasti s nízkou složitostí ze sekvence, která slouží jako dotaz do databáze. Oblast s nízkou složitostí znamená úsek sekvence, který je složen pouze z několika málo typů elementů. Takové úseky by mohly způsobovat zvýšení

skóre, které by pak vedlo k tomu, že by program ignoroval některé zajímavé sekvence z databáze, čímž by se snížila citlivost vyhledávání. Pro filtraci těchto úseků s nízkou složitostí se v nástroji BLAST využívají programy SEG (pro proteinové sekvence) a DUST (pro sekvence nukleotidů).

2. Vytvoří se seznam všech slov délky k, která obsahuje dotazovaná sekvence. Pro proteiny obvykle $k = 3$.

3. Ke slovům z druhého kroku se vytvoří seznam všech alternativních slov. Ke každému alternativnímu slovu je vypočteno skóre podobnosti vzhledem k původnímu slovu (např. pomocí matice BLOSUM62[22]) a jsou ponechány pouze ta alternativní slova, která mají skóre vyšší, než zadaný práh.

4. Ze slov z kroků 2 a 3 (slova s nejvyšším skóre) je vytvořen efektivní vyhledávací strom, který urychlí porovnávání se sekvencemi v databázi.

5. Databáze sekvencí je porovnána na přesnou shodu se slovy s nejvyšším skóre.

6. Přesné shody se slovy s nejvyšším skóre jsou rozšířeny na delší úseky. V nejnovější verzi nástroje BLAST jsou úseky rozšiřovány tak, že nejdříve se spojí dva úseky s přesnou shodou, které leží v pomyslné matici algoritmu Smith-Waterman na stejné diagonále. Potom je takto spojený úsek rozšířen na obě strany přesným algoritmem Smith-Waterman. Rozšiřování pokračuje do té doby, dokud skóre neklesne pod nastavený práh.

7. Vytvoří se seznam všech rozšířených úseků, které mají dostatečně velké skóre a zjistí se jejich statistická významnost, tzv. E-value, podle Gumbelovy funkce rozložení extrémních hodnot (Extreme Value Distribution). Pro představu:

$$p(S \geq x) = 1 - \exp\left(-e^{-\lambda(x-\mu)}\right) \tag{4.1}$$

$$E = 1 - e^{-p(s \geq x)D} \tag{4.2}$$

   kde parametry λ a μ jsou empiricky zjištěny funkční aproximací (závisí na použité matici porovnání a parametr D značí počet sekvencí v prohledávané databázi.

8. Hledají se skupiny konzistentních úseků, které na sebe navazují a spojují se ve větší celky.

9. Na výstupu se zobrazí každý výskyt, jehož E-value je menší než zadaný práh.

Speciální variantu nástroje BLAST představuje program PSI-BLAST, který je vhodný pro vyhledávání vzdálených příbuzných proteinů. Funguje v několika iteracích. V první iteraci jsou všechny nalezené proteinové sekvence zkombinovány do společného profilu (vícenásobného zarovnání), který shrnuje významné rysy těchto sekvencí. V druhé iteraci je jako dotaz pro vyhledávání položen právě tento společný profil, což vede k tomu, že je nalezeno více (vzdálenějších) sekvencí. Z výsledků další iterace je opět vytvořen společný profil, který bude opět sloužit jako vstup do další iterace. Tím, že PSI-BLAST zahrnuje do dalších fází výsledky z předchozích iterací, je o mnoho citlivější v hledání vzdálených evolučních vztahů mezi proteiny než standardní vyhledávání nástrojem BLAST.

PatternHunter

Nástroj PatternHunter[32] slouží ke stejnému účelu jako BLAST. Oproti němu, ale v úvodní fázi nepoužívá souvislá slova o délce k, ale více slov s různými rozestupy. Algoritmus staví na tom, že citlivost vyhledávání řetězců velmi ovlivňuje velikost mezery mezi sousedními slovy. Pokud použijeme dlouhá slova, nenajdeme izolované podobnosti (homologie). Na druhou stranu, pokud použijeme krátká slova (jako např. v nástroji BLAST), nalezneme spoustu víceméně náhodných výskytů, které zpožďují výpočet. PatternHunter nabízí jemnou rovnováhu mezi oběma variantami tím, že v úvodní fázi vyhledává kratší slova s optimální mezerou mezi nimi.

Podobně jako je BLAST mnohonásobně rychlejší, než použití algoritmu Smith-Waterman pro stejnou úlohu, je i PatternHunter mnohonásobně rychlejší než BLAST. Nutno ale podotknout, že za dobu své existence zatím nedokázal výrazným způsobem překonat popularitu a rozšířenost nástroje BLAST.

Identifikace příbuzných proteinů s využitím sekundární struktury

Tento zajímavý přístup k hledání příbuzných proteinů vychází z předpokladu, že struktura proteinu je více evolučně konzervovaná nežli samotná sekvence aminokyselin. Autoři Geourjon a spol. zkoumali možnosti identifikace vzdálených příbuzných proteinů, tedy v oblasti nízké vzájemné identity, kde u ostatních nástrojů typu BLAST nebo PatternHunter mluvíme již o oblasti šumu, ve které může být příliš mnoho falešných výskytů. Ukázali, že i mezi proteiny s identitou menší než 20 % lze na základě podobnosti jejich sekundární struktury (ať už známé, nebo predikované) identifikovat příbuzné proteiny[20].

HHblits

HHblits[45] je nástroj určený taktéž pro hledání homologních sekvencí. Zásadně se však liší v použitém matematickém modelu, který využívá pro vyhledávání. HHblits vygeneruje pro dotazovací sekvenci (nebo vícenásobné zarovnání) profile-HMM (profile Hidden Markov Model) a iterativně prohledává databázi předgenerovaných profile-HMM. Databáze profile-HMM se generuje z běžně dostupných databází (například UniProt, GeneBank, PDB apod.). Sekvence z těchto databází jsou shlukovány a pak je pro každý shluk vygenerován profile-HMM. Podobně jako PSI-BLAST funguje i HHblits iterativně. Po první iteraci jsou všechny významně podobné sekvence zahrnuty do konstrukce dotazovacího profile-HMM pro další iteraci.

Ve srovnání s nástrojem PSI-BLAST je HHblits rychlejší, až dvakrát citlivější a produkuje přesnější výsledné zarovnání, ale je vhodný spíše pro hledání vzdálených homologů.

4.2 Metody pro predikci funkce proteinu

Metody pro predikci funkce proteinu se snaží řešit problém, jak neznámému proteinu přiřadit jeho pravděpodobnou funkci. Existuje několik přístupů, které se snaží tento problém řešit s menší či větší úspěšností.

Metoda založená na homologii

Vychází z předpokladu, že podobné sekvence jsou zpravidla homologní (mají společného předka)[43], a proto můžeme usuzovat, že sekvenčně podobné proteiny budou mít stejnou funkci. Toto však neplatí vždy. Je možné nalézt dva proteiny, která mají velice podobnou sekvenci a přesto mají velice rozdílnou funkci. Například proteiny Gal1 a Gal3 z organismu kvasinek vykazují 73% identitu a 92% podobnost, přesto ale mají různou funkci. Gal1 plní funkci galaktokinázy a Gal3 funkci transkripčního faktoru[24]. Neexistuje tedy žádný pevný práh podobnosti sekvencí, pomocí kterého bychom dokázali bezpečně určit, mají-li dva proteiny stejnou funkci.

Funkci proteinu však můžeme dále ověřit podrobnější analýzou. Podaří-li se nám nalézt homologní protein se známou funkcí a se známou strukturou, můžeme pro náš vstupní protein určit terciární strukturu pomocí tzv. homologního modelování (známá struktura poslouží jako templát) s tím, že pozice důležitých katalytických reziduí, které jsou klíčové pro funkci proteinu, zafixujeme. Pokud se nám podaří sestrojit takovýto homologní model s malou chybou, je vysoce pravděpodobné, že nalezená homologní sekvence bude mít i stejnou funkci. Tohoto poznatku je využito i při návrhu nástroje, který je výstupem této práce.

Nejznámější nástroj pro predikci terciární struktury proteinu založený na homologii je pak MODELLER[19]. Ten implementuje techniku inspirovanou nukleární magnetickou rezonancí známou jako metoda *splnění prostorových omezení* (volně přeloženo), která využívá množinu geometrických kritérií pro vytvoření funkce hustoty rozdělení pravděpodobnosti pro pozici každého atomu v proteinu. MODELLER poskytuje i základní podporu pro tzv. *ab initio* modelování úseků proteinové sekvence tvořících smyčky, které jsou obvykle vysoce variabilní i mezi navzájem homologními sekvencemi.

Metoda založená na sekvenčním motivu

Metoda založená na sekvenčním motivu předpokládá, že je možné v neznámé sekvenci vyhledat známý sekvenční motiv. Ke známým motivům pak lze dohledat anotace naznačující funkce, které obvykle plní proteiny s tímtéž sekvenčním motivem. Jednou z databází sekvenčních motivů je například PROSITE (Database of Protein Domains, Families and Functional Sites)[50].

Metoda založená na strukturním zarovnání

Jak již bylo zmíněno, struktura proteinu je obecně více konzervovaná než jeho sekvence, proto strukturní podobnost dvou proteinů je dobrým indikátorem toho, že by mohly mít i stejnou funkci. Metoda založená na strukturním zarovnání proto využívá znalosti struktury proteinu s neznámou funkcí a snaží se nalézt strukturně podobné proteiny např. v databázi PDB[9]. Pokud struktura proteinu s neznámou funkcí není dostupná, mohou se některé nástroje nejdříve pokusit strukturu proteinu predikovat a teprve poté vyhledávat v databázi známých struktur. Pro porovnání struktur proteinu se používají nástroje pro strukturní zarovnání a hodnotí se odchylka mezi strukturami.

Kapitola 5

Návrh nástroje

Na nejvyšší úrovni návrhu lze nástroj popsat jako tři navazující logické bloky – hledání homologních sekvencí, filtrace výsledků a *in silico* screening zbylých sekvencí (viz obrázek 5.1). Pojem screening je zde použit v širším významu, než je chápán např. v rámci lékařství, kde obvykle značí vyšetřování předem definované skupiny lidí za účelem vyhledávání chorob v jejich časných stádiích, kdy pacient ještě nemá potíže a příznaky[58]. V tomto textu je screening použit ve smyslu prověřování, či vyšetřování vlastností všech nalezených enzymů, podobně jako se pojmu užívá i v laboratorním prostředí.

Účelem bloku hledání homologních sekvencí je identifikovat v databázi velké množství sekvencí podobných těm, které byly použity jako vstupy. Cílem filtrace je na základě znalosti pozic a typů katalytických reziduí odfiltrovat všechny takové sekvence, které požadovaná rezidua neobsahují. Nakonec smyslem fáze screeningu je dozvědět se co největší množství relevantních informací o zbylých sekvencích, které usnadní třídění výsledků a výběr vhodných kandidátů pro experimentální ověření v laboratoři.

Každý logický blok se skládá z několika dílčích navazujících fází (viz obrázek 5.2). V následujícím textu jsou jednotlivé fáze blíže představeny a pro každou jsou shrnuty nutné vstupy a výstupy.

Prezentovaný návrh vznikl ve spolupráci s Loschmidtovými laboratořemi v Brně[30], které mají v oboru proteinového inženýrství dlouholeté zkušenosti a patří mezi špičková evropská pracoviště.

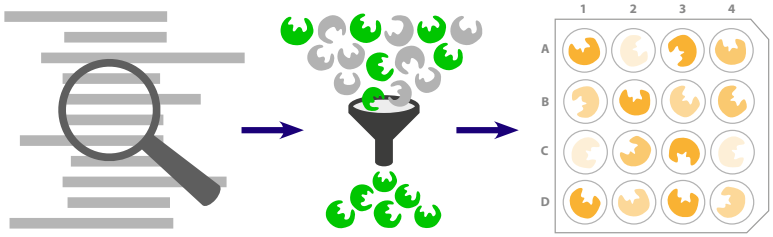
5.1 Povinné vstupy

Známé sekvence – sekvence známých proteinů s požadovanou funkcí, například aktuálně experimentálně potvrzené bez mutantů.

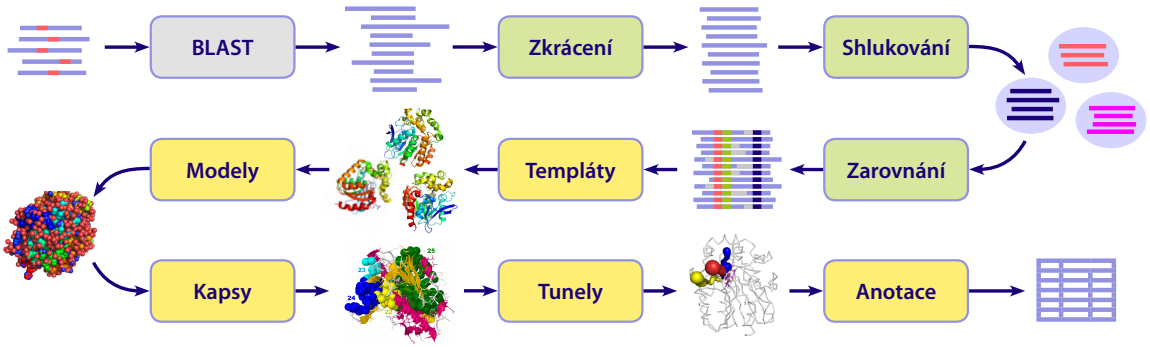
Doplňující sekvence – sekvence putativních proteinů, které ještě nejsou experimentálně potvrzené, ale je pravděpodobné, že budou mít požadovanou funkci.

Sekvence pro vyhledávání – sekvence proteinů, které budou použity ve fázi prohledávání databáze všech známých proteinů. Optimálně aspoň jedna sekvence pro každou podrodinu. Pokud má jako vyhledávací sekvence sloužit např. vícedoménový protein a uživatele zajímá jen jedna doména, předpokládá se, že tato sekvence bude zkrácena takovým způsobem, aby obsahovala jen tu doménu, o kterou má zájem.

Specifikace katalytických reziduí – specifikace pozic a typů katalytických reziduí pro některé ze *známých sekvencí*.



Obrázek 5.1: *Abstraktní pohled na návrh vyhledávacího algoritmu* – 1. hledání homologních sekvencí, 2. filtrace, 3. in silico screening nalezených sekvencí.



Obrázek 5.2: *Schéma dílčích kroků algoritmu* – podbarvení jednotlivých fází odkazuje na abstraktní pohled z obrázku 5.1. Podrobnější popis naleznete v příslušných kapitolách – BLAST (5.2), zkrácení (5.3), shlukování (5.4, 5.5), zarovnání (5.6), templáty (5.7), modely (5.8), kapsy (5.9), tunely (5.10), anotace (5.11).

Hlavní templát – struktura (ve formátu PDB), se kterou budou strukturně zarovnány všechny homologní modely.

Vlastní sekvence – sekvence pro vlastní struktury, které nejsou v PDB databázi, ale bylo by vhodné je zařadit mezi potenciální templáty pro homologní modelování.

Specifikace katalytických reziduí ve vlastních sekvencích – definice katalytických pozic pro vlastní sekvence, potažmo struktury.

Vlastní struktury – PDB a mmCIF soubory pro vlastní struktury.

5.2 Prohledávání databáze sekvencí známých proteinů

Vstupy:

- BLAST databáze **NR** – výchozí neredundantní databáze pro vyhledávání proteinových sekvencí nástrojem BLAST, která obsahuje všechny sekvence proteinů z databází RefSeq[42], Swiss-Prot[5], PDB[9], PIR[7] a PRF spolu s překlady všech genů z databáze GeneBank do proteinových sekvencí.
- Sekvence pro vyhledávání, známé sekvence.

V této fázi se provede vyhledávání příbuzných proteinových sekvencí z databáze a jako dotaz se použijí zvlášť všechny sekvence pro vyhledávání.

Výstupy:

- Putativní sekvence získané vyhledáváním v databázi spolu se známými sekvencemi (viz 5.1).

5.3 Zkrácení sekvencí a párové přiložení

Vstupy

- Putativní sekvence, sekvence pro vyhledávání, známé sekvence.

Všechny sekvence identifikované pomocí homologního vyhledávání je nutné zkrátit tak, aby zhruba odpovídaly sekvencím pro vyhledávání. Tímto budou z nalezených sekvencí odstraněny domény, které se nevyskytují v sekvencích pro vyhledávání. Pokud by odstraněny nebyly, mohly by nastat problémy při shlukování). Na závěr této fáze se provede párové přiložení (globální zarovnání) všech zkrácených sekvencí se všemi známými sekvencemi, které se využije při shlukování k rozpoznání známých nebo jim blízkých sekvencí.

Výstupy

- Zkrácené sekvence – identifikované sekvence zkrácené tak, aby obsahovaly jen cílovou doménu.
- Párové přiložení se známými sekvencemi – porovnání zkrácených sekvencí se všemi známými sekvencemi. Obsahuje jen dvojice s identitou vyšší než 90 %.

5.4 Konstrukce matice vzdáleností

Vstupy

- Zkrácené sekvence.

Nejprve se provede vzájemné globální párové přiložení všech zkrácených sekvencí mezi sebou a připraví se matice párových vzdáleností ve formátu vhodném pro shlukování, tj. sekvence budou číslovány od jedné do počtu sekvencí a každá dvojice se bude v matici vyskytovat pouze jednou.

Výstupy

- Matice vzdáleností mezi všemi zkrácenými sekvencemi ve formátu vhodném pro shlukování.
- Transformační soubor mapující GI číslo (Gene Identification) na pořadové číslo sekvence v matici vzdáleností.

5.5 Shlukování

Vstupy

- Putativní sekvence, matice vzdáleností, transformační soubor.
- Párové přiložení se známými sekvencemi.

Shlukování je nutným krokem před výpočtem vícenásobného zarovnání (MSA) nalezených sekvencí, jinak bychom nedosáhli kvalitního výsledku, proto je provedeno hierarchické shlukování sekvencí dle matice párových vzdáleností. Získaný strom shluků je pak ořezán na různých úrovních a je vybráno takové rozdělení, které nejlépe pokrývá množinu identifikovaných sekvencí a ideálně v každém shluku je jedna ze známých nebo jim blízkých sekvencí (viz 5.1). Na závěr se vytvoří soubory obsahující sekvence patřící do daných shluků, což v další fázi umožní provést konstrukci profile-profile MSA.

Výstupy

- Soubory s informacemi o sekvencích patřících do jednotlivých shluků při dané úrovni řezu hierarchickým stromem.
- Strom shluků – hierarchie sekvencí, resp. shluků.

5.6 Konstrukce vícenásobného zarovnání

Vstupy

- Soubory s informacemi o shlucích z předchozí fáze, strom shluků, specifikace katalytických reziduí.

Pro každý shluk z předchozí fáze se vytvoří MSA. Dále podle hierarchie, která je dána stromem shluků, se postupně provede profile-profile MSA a to tak, že se v každém kroku budou vybírat dva nejbližší shluky. Všechny sekvence, které v hlavičce obsahují některé z klíčových slov synthetic, vector, apod., jsou označeny jako umělé.

V získaném MSA se podle specifikace katalytických reziduí určí jejich pozice a pro každou sekvenci se zkontroluje, jestli obsahuje specifikovaná katalytická rezidua. Sekvence, které neobsahují všechna potřebná rezidua (např. z důvodu delece, či substituce) jsou označeny jako nekompletní nebo degenerované a smazány. Tento krok vede k tomu, že z velkého množství sekvencí zůstanou takové, které pravděpodobně budou tvořit podobný protein s mírně modifikovanou funkcí, protože obsahují kompletní sadu katalytických reziduí.

Výstupy

- Identifikované sekvence – výsledná množina identifikovaných sekvencí bez umělých, nekompletních nebo degenerovaných sekvencí.
- Vícenásobné zarovnání – MSA výsledné množiny identifikovaných sekvencí.
- Katalytická rezidua – zjištěná katalytická rezidua pro sekvence z výsledné množiny na základě MSA.
- Pozice katalytických reziduí.

5.7 Vyhledávání templátů

Vstupy

- Identifikované sekvence, vícenásobné zarovnání, specifikace katalytických reziduí, pozice katalytických reziduí, vlastní sekvence, specifikace katalytických pozic ve vlastních sekvencích, vlastní struktury.

Z identifikovaných sekvencí se vyberou takové, pro které existuje záznam v PDB databázi struktur. K těmto sekvencím se přidají všechny vlastní sekvence (tj. sekvence vlastních proteinů s určenou strukturou, které dosud nejsou dostupné v PDB databázi) a vytvoří se vlastní databáze sekvencí.

Ve výsledném vícenásobném zarovnání se stanoví konsensuální začátek a konec sekvencí se známou strukturou. Např. začátek se stanoví na pozici MSA, kde má aspoň 50 % sekvencí nějaké reziduum (a ne mezeru), obdobně pro konec. Vícenásobné zarovnání se pak ořeže podle konsensuálního začátku a konce. Ořezané sekvence budou použity pro konstrukci homologních modelů a tím pádem pro všechny následné strukturní analýzy. Pro každou sekvenci se uloží počet odstraněných reziduí (velikost domény) ze začátku a konce a určí se katalytická rezidua a jejich pozice.

Každá ořezaná sekvence je použita jako dotaz pro vyhledávání ve vlastní databázi (viz výše). Nakonec se z výsledků vyhledávání určí vhodný templát pro všechny sekvence k následnému homologním modelování.

Výstupy

- Ořezané zarovnání – vícenásobné zarovnání putativních sekvencí ořezané dle určeného konsenzu začátku a konce sekvencí.
- Katalytická rezidua pro ořezané sekvence.
- Pozice katalytických reziduí v ořezaných sekvencích.
- Pozice katalytických reziduí v templátech – definice katalytických pozic pro všechny potenciální templáty.
- Odstraněné domény – počet N-terminálních a C-terminálních reziduí odstraněných z jednotlivých sekvencí při vytváření ořezaného zarovnání.

5.8 Homologní modelování

Vstupy

- Katalytická rezidua pro ořezané sekvence a jejich pozice v potenciálních templátech.

Všechny potenciální templáty se vhodným způsobem ohodnotí. Do ohodnocení templátu se zahrne kvalita jeho párového přiložení s modelovanou sekvencí, správnost pozic a typů katalytických reziduí i rozlišení templátu. Nakonec se vybere templát s nejlepším ohodnocením a použije se ke konstrukci homologního modelu.

Výstupy

- Homologní modely pro cílové sekvence zkonstruované podle nalezených templátů.
- Párová přiložení sekvencí templátů s modelovanými sekvencemi.
- Informace o templátech použitých pro homologní modelování cílových sekvencí.

5.9 Strukturní zarovnání a CASTp analýza

Vstupy

- Hlavní struktura, se kterou mají být zarovnány všechny homologní modely.
- Katalytická rezidua pro ořezané sekvence.

Každý homologní model se strukturně zarovná s hlavní strukturou a na zarovnaném modelu se spustí analýza kapes a dutin.

Výstupy

- Katalytické kapsy – informace o katalytických kapsách identifikovaných v homologních modelech.
- Výsledky CASTp analýzy.

5.10 Hledání tunelů

Vstupy

- Strukturně zarovnané homologní modely.

Provede se výpočet a shlukování tunelů ve všech zkonstruovaných a zarovnaných homologních modelech. Homologní modely jsou analyzovány jako jedna trajektorie, což umožní vzájemné porovnání tunelů a jejich shlukování.

Výstupy

- Charakteristika tunelů.

5.11 Databáze Taxonomy, Bioproject a Pfam

Vstupy

- Identifikované putativní sekvence.

Z NCBI databází Taxonomy[36, 35] a Bioproject[34] se pro každou putativní sekvenci vyextrahuje informace o hostitelském organismu, ze kterého daná sekvence pochází, informace o salinitě, optimální teplotě a rozmezí teplot. Pro každou putativní sekvenci se provede sekvenční prohledání Pfam databáze[17] a vyextrahuje se informace o obsažených doménách.

Výstupy

- Informace o hostitelském organismu.
- Pfam anotace domén putativních sekvencí

Kapitola 6

Implementace nástroje

V této kapitole jsou rozvedeny technické detaily týkající se implementace navrženého workflow z kapitoly 5. První část se věnuje použitému aplikačnímu frameworku a jeho klíčovým komponentám, druhá část je zaměřena na implementaci dílčích modulů a v neposlední řadě na související datové soubory.

6.1 Framework Loschmidtových laboratoří

Loschmidtovy laboratoře, v jejichž spolupráci návrh algoritmu vznikal, vedou vývoj hned několika bioinformatických nástrojů. Mezi nejznámější z nich patří např. CAVER[28] – software pro identifikaci a charakterizaci přístupových tunelů v makromolekulách, HotSpot Wizard[39] – webový server pro automatickou identifikaci míst vhodných k mutaci, nebo nejnovější PredictSNP[8] – prediktor vlivu škodlivosti aminokyselinových mutací. Tyto zkušenosti vyústily v implementaci vlastního bioinformatického frameworku, který usnadňuje údržbu stávajících a vývoj nových aplikací v jazyce Java.

Framework je zaměřen především na aplikace typu HotSpot Wizard nebo PredictSNP, které lze charakterizovat jako webové služby. Typická interakce s nimi zahrnuje následující kroky: uživatel zadá požadované vstupy a odešle úlohu, server zařadí výpočet s danou konfigurací do fronty, ve vhodném okamžiku spustí výpočet a po jeho doběhnutí oznámí uživateli výsledky. Funkcionalita frameworku se omezuje pouze na podporu implementace serverové části služby, např. plánování úloh, konfigurace, či spouštění nástrojů.

Komponenty frameworku

Framework Loschmidtových laboratoří se skládá ze dvou samostatných částí – Core (jádra) a Commons (knihovny). *Jádro* zajišťuje základní provozní funkce výpočetní služby – pro představu je uveden seznam některých komponent jádra a jejich funkce.

- `module` – základní komponenta. Každý výpočetní modul (např. BLAST vyhledávání, shlukování, apod.) musí být specializací této komponenty, aby mohl využívat služeb jádra – ať už např. komponentu `storage` nebo centrální logovací systém.
- `storage` – řídí přístup k datovým souborům modulů a zajišťuje oddělený prostor pro každou úlohu.
- `database` – komunikace s databází. Nástroje s webovým rozhraním skrze databázi přijímají úlohy ke zpracování a do databáze nahrávají výsledky výpočtů.

- `configuration` – čtení a správa konfigurace úloh a spouštěných modulů. Jádro přijímá konfiguraci ve formátu XML.
- `dependency` – komponenta pro specifikaci závislostí mezi moduly – určuje možné pořadí spouštění modulů a identifikuje úlohy, které lze spouštět paralelně.
- `queue` – manažer fronty úloh.
- `calculation` – abstrakce pro právě běžící výpočet na některém z pracovních vláken jádra. Uchovává stav výpočtu, čas běhu, apod.
- `job` – souhrnný datový kontejner pro výpočetní úlohy. Každá úloha má přidělen unikátní identifikátor v rámci celého systému, aby bylo možné se dotazovat na stav výpočtu nebo výsledky dané úlohy.

Knihovna obsahuje všechny ostatní algoritmy a komponenty, které nesouvisí se zajištěním běhu a prostředí výpočetních úloh a nejsou specifické pro konkrétní projekt, případně službu. V následujícím seznamu je uvedeno několik nevýznamnějších komponent.

- `protein` – datová reprezentace proteinu v rámci frameworku. Obsahuje funkcionalitu pro načítání a ukládání proteinových struktur (PDB a mmCIF souborů).
- `sequence` – komponenta usnadňující práci s proteinovými sekvencemi. Podporuje např. načítání a ukládání souborů ve FASTA formátu, nebo čtení vícenásobných zarovnání.
- `tool` – obsahuje abstrakce pro řadu bioinformatických nástrojů a sjednocuje způsob jejich spouštění. Mezi zahrnuté nástroje patří BLAST, CASTp, ClustalΩ, FastTree, Rate4Site a mnohé další. Množina nástrojů byla v rámci práce na tomto projektu dále rozšířena.

Adresářová struktura pracovního adresáře

Pro správný běh frameworku je vyžadována specifická struktura pracovního adresáře aplikace – musí obsahovat následujících šest podsložek s odpovídajícím obsahem.

- `cache` – zpočátku prázdná složka. Moduly do ní mohou ukládat data, která by se mohla opakovaně hodit a trvá nezanedbatelnou dobu je vypočítat nebo jinak získat. Této možnosti využívá např. komponenta *PdbService* použitá v modulu *PdbBlast* pro stahování PDB souborů (viz 6.3.6).
- `conf` – obsahuje konfigurační soubory modulů nebo částí frameworku. Např. konfigurační soubory bioinformatických nástrojů z komponenty `tool` určují absolutní cestu ke spustitelnému souboru a parametry příkazové řádky v různých kontextech použití (viz příloha A).
- `data` – složka s perzistentními daty vhodná k umístění databází, které z pohledu aplikace není třeba často aktualizovat. Např. v rámci toho projektu se jedná o BLAST databázi NR, databázi BioProject nebo soubory projektu Taxonomy.
- `jobs` – prostor, kam jádro frameworku ukládá informace týkající se přijatých úloh, ať už se jedná o nastavení úlohy v XML formátu nebo o přiřazené logovací soubory z průběhu výpočtu.

- `software` – složka určená pro spustitelné soubory bioinformatických nástrojů.
- `temp` – pracovní adresář pro běžící úlohy. Jádro přidělí každé úloze unikátní složku v tomto adresáři, jejíž název je shodný s identifikátorem úlohy – v této složce jsou pak dále pracovní adresáře jednotlivých modulů dané úlohy a také adresář `shared`, skrze který mohou moduly vzájemně sdílet soubory. Sdílení dat na úrovni celých úloh je pak možné pouze skrze adresář `data` nebo `cache`.

6.2 Rejstřík sdílených souborů

Vzhledem k tomu, že návrh je koncipován jako *workflow*, tzn. jako spojení existujícího bioinformatického software, a jedinou možností, jak s tímto software komunikovat je skrze soubory a parametry příkazové řádky, pracuje výsledná implementace s poměrně velkým množstvím různých souborů a složek.

Proto, aby se mohl čtenář snadněji orientovat v tom, jaká data konkrétní důležité soubory obsahují, následuje stručný popis všech souborů, které jsou vytvářeny ve složce `shared` pracovního adresáře úlohy, a to v pořadí fází, z nichž pocházejí, nebo kde jsou poprvé potřeba, dle kapitoly 5. V hranatých závorkách je případně uveden odkaz na modul, který soubor vytváří. Není-li uvedeno nic, pak je soubor vstupem od uživatele.

- `queries.fas` – sekvence proteinů, které budou použity jako vstup vyhledávání podobných sekvencí v modulu 6.3.1. Ideálně by měl obsahovat z každé podrodiny zkoumaného typu enzymu jednu sekvenci, aby ve výsledcích byly zastoupeny sekvence ze všech podrodin.
- `known_sequences.fas` – sekvence známých proteinů, např. experimentálně potvrzených. Využívá se hned v několika modulech, nejvýraznější roli hraje při identifikaci relevantních shluků ve fázi shlukování (viz 6.3.4).
- `seqs_blast.fas` [6.3.1] – identifikované sekvence podobných proteinů z databáze NR obohacené o sekvence známých proteinů ze souboru `known_sequences.fas`.
- `seqs_blast_shortened` [6.3.2] – zkrácené sekvence ze souboru `seqs_blast.fas` tak, aby neobsahovaly nadbytečné domény, které by mohly snižovat kvalitu shlukování.
- `usearch_globalDB_known` [6.3.2] – porovnání sekvencí z `seqs_blast.fas` se známými sekvencemi z `known_sequences.fas`. Obsahuje jen dvojice s vyšší než 90% identitou. Formát: známá sekvence; identifikovaná sekvence; procentuální sekvenční identita; délka známé sekvence; délka identifikované sekvence.
- `usearch_dbGlobal.edges.gz` [6.3.3] – matice vzájemných párových vzdáleností mezi všemi zkrácenými sekvencemi z `seqs_blast_shortened` ve formátu vhodném pro UPGMA shlukování. Formát: pořadové číslo první sekvence; pořadové číslo druhé sekvence; vzdálenost. Komprimováno algoritmem GZIP.
- `transformation_file` [6.3.3] – soubor mapující GI číslo sekvence na pořadové číslo sekvence. Formát: GI číslo; pořadové číslo sekvence – tj. číslo v UPGMA matici.
- `[cutoff]_[clusterid]_identified_targets.fas` [6.3.4] – putativní sekvence identifikované při oříznutí UPGMA stromu na prahové hodnotě `[cutoff]` patřící do shluku číslo `[clusterid]`.

- `cluster_tree` [6.3.4] – hierarchie sekvencí resp. shluků sekvencí vytvořená nástrojem MC-UPGMA podle matice vzdáleností `usearch_dbGlobal.edges.gz`.
- `catalytic_aas.txt` – specifikace katalytických reziduí pro vybrané známé sekvence z `known_sequences.fas`.
- `identified_targets.fas` [6.3.5] – finální množina identifikovaných putativních sekvencí bez umělých, nekompletních a degenerovaných sekvencí.
- `msa_identified_targets.fas` [6.3.5] – vícenásobné zarovnání finální množiny putativních sekvencí z `identified_targets.fas` konstruované podle stromu shluků `cluster_tree`.
- `catalytic_pentads.txt` [6.3.5] – identifikovaná katalytická rezidua pro sekvence z `identified_targets.fas` na základě `msa_identified_targets.fas`. Formát: GI číslo; důvod odstranění; jednotlivá katalytická rezidua.
- `catalytic_positions.txt` [6.3.5] – rezidua identifikovaných sekvencí odpovídající katalytickým pozicím známých sekvencí podle `msa_identified_targets.fas`. Formát: GI číslo; důvod odstranění; rezidua odpovídající jednotlivým katalytickým pozicím.
- `own_structures.fas` – sekvence pro vlastní struktury, které mají být zařazeny mezi potenciální templáty pro homologní modelování, ale nejsou dostupné v PDB databázi.
- `o_catalytic.txt` – definice pozic katalytických reziduí v sekvencích vlastních struktur z `own_structures.fas`.
- `o_[nn].cif` – mmCIF (macromolecular Crystallographic Information File) soubory pro vlastní struktury. `[nn]` zastupuje dvouciferné identifikační číslo struktury, které je odkazováno v rámci hlavičky v souboru `own_structures.fas`.
- `o_[nn].pdb` – PDB (Protein Data Bank) soubory pro vlastní struktury. Význam `[nn]` je stejný jako výše.
- `msa_identified_targets_cut.fas` [6.3.6] – MSA finální množiny identifikovaných putativních sekvencí z `msa_identified_targets.fas` ořezané dle určeného konsenzu začátku a konce sekvencí se známou strukturou.
- `catalytic_pentads_cut.txt` [6.3.6] – identifikovaná katalytická rezidua pro ořezané sekvence z `msa_identified_targets_cut.fas`.
- `catalytic_positions_cut.txt` [6.3.6] – rezidua ořezaných sekvencí odpovídající katalytickým pozicím známých sekvencí podle `msa_identified_targets_cut.fas`.
- `catalytic_positions_templates.txt` [6.3.6] – definice pozic katalytických reziduí pro všechny potenciální templáty. Formát: PDB ID; rezidua odpovídající jednotlivým katalytickým pozicím.
- `removed_domains.txt` [6.3.6] – počet N-terminálních a C-terminálních reziduí odstraněných ze sekvencí při vytváření ořezaného MSA. Formát: GI číslo; počet odstraněných reziduí N-terminálně; počet odstraněných reziduí C-terminálně.

- `modeller/[n]/[m]/` [6.3.6] – složka obsahující úlohy pro MODELLER a jejich výsledky. `[n]` zastupuje pořadové číslo sekvence z `identified_targets.fas` a `[m]` pořadové číslo potenciálního templátu.
- `modeller/template_db/` [6.3.6] – strukturní soubory PDB a mmCIF pro možné templáty.
- `catalytic_pocket.txt` [6.3.7] – informace o katalytických kapsách identifikovaných v homologních modelech z `modeller/[n]/`. Formát: GI číslo; ID kapsy; plocha; objem; PDB kód templátu pro homologní modelování.
- `selected_template_info.txt` [6.3.7] – informace o templátech použitých pro homologní modelování identifikovaných sekvencí. Formát: GI číslo; pořadové číslo složky `[m]` s párovým přiložením v rámci úlohy pro MODELLER; PDB kód vybraného templátu; skóre vyjadřující kvalitu homologního modelu.
- `tunnels/` [6.3.7] – složka se vstupy pro výpočet tunelů, obsahuje mj. `config.txt` – konfigurační soubor pro CAVER a složku `pdbs` se vstupními PDB soubory pro analýzu tunelů – tj. zarovnané homologní modely s modifikovanými čísly atomů.
- `cavities/[n]/` [6.3.7] – výsledky CASTp analýzy.
- `extremophilicity.txt` [6.3.8] – pro sekvence z `identified_targets.fas` podle GI čísla poskytuje informace o hostitelském organismu. Pokud je určité GI číslo asociováno s více organismy, je každý záznam uveden na zvláštním řádku. Formát: GI číslo; organismus; říše (A – archea, B – bacteria, E – eukaryota, V – viruses and viroids, U – unclassified and other); salinita; optimální teplota; rozmezí teplot.
- `pfam_parsed.txt` [6.3.9] – Pfam anotace sekvencí z `identified_targets.fas`. Pokud je v sekvenci identifikováno více domén, je sekvence uvedena na více řádcích. Formát: GI číslo; přístupový kód identifikované domény do databáze Pfam; název domény; začátek a konec domény v rámci sekvence; E-value domény.

6.3 Moduly

Tato podkapitola obsahuje popis fungování a implementace jednotlivých kroků navrženého algoritmu jako modulů v rámci frameworku Loschmidtových laboratoří. U každého modulu je ve stručné hlavičce uvedeno, kde lze nalézt související zdrojové kódy, jaké nástroje při svém běhu využívá, co vyžaduje na vstupu, a které výstupní soubory produkuje.

6.3.1 HomologySearch

Umístění:	`loschmidt.hts.module.homology_search`
Nástroje:	PSI-BLAST[12], BLASTdbcmd[12]
Vstupy:	NR databáze, `queries.fas`, `known_sequences.fas`
Výstupy:	`seqs_blast.fas`

Soubor `queries.fas` je rozdělen na dílčí sekvence a pro každou z nich se paralelně spustí PSI-BLAST prohledávání databáze NR s následujícími parametry – počáteční práh E-value:

10^{-20}, práh E-value pro zahrnutí sekvence do další iterace: 10^{-20}, počet iterací: 2, maximální množství výsledků: 20 000, výstupní formát: XML. Parametry jsou zvoleny s ohledem na to, aby výsledky obsahovaly spíše příbuznější sekvence, nežli vzdálené homology.

Z výstupních XML souborů se extrahují přístupové identifikátory všech sekvencí a sjednotí se do jednoho seznamu bez duplicit. Mohlo se totiž přihodit, že některá ze sekvencí byla nalezena vícekrát, protože dílčí vyhledávání jsou na sobě nezávislá.

Seznam přístupových identifikátorů se použije jako vstup nástroje BLASTdbcmd, který z databáze NR vybere odpovídající sekvence a ve formátu FASTA je zapíše do souboru `seqs_blast.fas`.

Nakonec se k souboru `seqs_blast.fas` připojí i sekvence z `known_sequences.fas` s vhodně upravenými hlavičkami. Jejich přítomnosti se využije posléze v modulu provádějícím shlukování k rozeznání relevantních shluků (viz 6.3.4).

6.3.2 SeqShortening

Umístění:	`loschmidt.hts.module.seqshortening`
Nástroje:	USEARCH[15]
Vstupy:	`queries.fas`, `known_sequences.fas`, `seqs_blast.fas`
Výstupy:	`seqs_blast_shortened`, `usearch_globalDB_known`

Sekvence ze souboru `queries.fas` se zarovnají pomocí nástroje USEARCH se všemi identifikovanými sekvencemi z `seqs_blast.fas`. Minimální požadovaná identita je nastavena na 0 %, tzn. bez omezení, a je použit přesný algoritmus globálního zarovnání Needleman-Wunsch[37].

Z výsledného zarovnání se pro každou identifikovanou sekvenci extrahuje číslo první a poslední aminokyseliny, pro kterou existuje v zarovnání protějšek v nějaké sekvenci z `queries.fas`.

Aplikuje se parametr tolerance zkrácení – první a poslední aminokyselina se posune směrem k N-terminálnímu, resp. C-terminálnímu konci (např. původně určený začátek a konec na aminokyselinách 145 a 320 se při aplikaci tolerance 20 posune na aminokyseliny 125 a 340). Aminokyseliny před určenou začáteční aminokyselinou, resp. za poslední aminokyselinou se oříznou (tj. v příkladu výše: aminokyseliny 1–124 a 341–konec) a výsledek se zapíše do souboru `seqs_blast_shortened`.

Nakonec se zarovnají všechny identifikované sekvence z `seqs_blast.fas` se známými sekvencemi z `known_sequences.fas` – výstupem je `usearch_globalDB_known`. Minimální požadovaná identita je tentokrát nastavena na 90 %, což znamená, že program USEARCH vypíše pouze dvojce s vyšší než 90% identitou, čehož bude použito k rozeznání známých sekvencí v datové sadě identifikovaných sekvencí.

6.3.3 SimilarityMatrix

Umístění:	`loschmidt.hts.module.similaritymatrix`
Nástroje:	USEARCH[15]
Vstupy:	`seqs_blast_shortened`
Výstupy:	`usearch_dbGlobal.edges.gz`, `transformation_file`

Nástrojem USEARCH se vzájemně porovnají sekvence z `seqs_blast_shortened` mezi sebou – výstupem je soubor párových podobností `usearch_global_all` ve formátu: GI číslo 1. sekvence; GI číslo 2. sekvence; podobnost v rozsahu 0–100 %.

Následuje konverze párových podobností na matici vzdáleností vhodnou pro UPGMA shlukování. Především jsou odstraněny duplicity – soubor `usearch_global_all` obsahuje každou dvojici sekvencí dvakrát – tzn. jak podobnost mezi 1. a 2. sekvencí, tak i mezi 2. a 1. Dále je odstraněna zbytečná informace o porovnání stejných sekvencí mezi sebou. Sekvenční podobnost je zaokrouhlena na celé číslo a převedena na vzdálenost podle vztahu 100 − podobnost. GI čísla jsou převedena na pořadová čísla a výsledná matice vzdáleností je uložena do souboru `usearch_dbGlobal.edges.gz` ve formátu: pořadové číslo 1. sekvence; pořadové číslo 2. sekvence; vzdálenost v rozsahu 0–100 a komprimována algoritmem GZIP.

Záznam o převodu GI čísel je uložen do souboru `transformation_file` pro pozdější zpětné dohledání ve formátu: GI číslo; pořadové číslo.

6.3.4 Clustering

Umístění: `loschmidt.hts.module.clustering`

Nástroje: MC-UPGMA[29]

Vstupy: `usearch_dbGlobal.edges.gz`, `transformation_file`, `usearch_globalDB_known`, `seqs_blast.fas`

Výstupy: `[cutoff]_[clusterid]_identified_targets.fas`, `cluster_tree`,

Provede se hierarchické shlukování nástrojem MC-UPGMA podle matice párových vzdáleností `usearch_dbGlobal.edges.gz` – výstupem je strom shluků v souboru `cluster_tree`. Získaný strom se ořeže na různých prahových hodnotách (tzv. *cutoff*) z povoleného rozsahu.

Pro každý řez se zkoumají shluky obsahující známé sekvence nebo jim velice podobné podle `usearch_globalDB_known`, určí se jejich počet a počet sekvencí, které obsahují. K tomu je nutná zpětná transformace pořadových čísel sekvencí ve shlucích na původní GI čísla pomocí `transformation_file`.

Zahodí se řezy, které obsahují počet shluků neodpovídající povolenému rozsahu. Dále pro každou různou hodnotu počtu shluků ve zbývajících řezech, je ponechán vždy jen jeden řez a to takový, který byl proveden na maximální prahové hodnotě. Pokud např. existují řezy na prahových hodnotách 60, 61, 62 a 63 shodně s počtem shluků 3, je ponechán pouze řez na hodnotě 63.

Pro každý zbylý řez se vypočítá rozdíl mezi celkovým počtem sekvencí ve shlucích při dané prahové hodnotě a při hodnotě o jedno vyšší. Např. dejme tomu, že při hodnotě 74 jsou známé sekvence obsaženy ve 4 shlucích, které dohromady obsahují 700 sekvencí; při prahové hodnotě 75 jsou známé sekvence ve 3 shlucích, které ale opět dohromady obsahují 700 sekvencí – výsledný rozdíl je roven 0. Jako finální je vybrán takový řez, který má tento rozdíl nejnižší. Pokud více řezů má shodně nejnižší rozdíl, je vybrán ten, který byl vytvořen na základě nejvyšší prahové hodnoty.

Připraví se vstupy pro vícenásobné zarovnání – sekvence patřící do shluků obsahujících známou sekvenci podle finálního řezu jsou načteny ze souboru `seqs_blast.fas` a vytištěny do souborů tvaru `[cutoff]_[clusterid]_identified_targets.fas`, kde `[cutoff]` zastu-

puje prahovou hodnotu finálního řezu a `[clusterid]` pořadové číslo shluku s aspoň jednou známou sekvencí.

6.3.5 Msa

Umístění:	`loschmidt.hts.module.msa`
Nástroje:	ClustalΩ[49]
Vstupy:	`[cutoff]_[clusterid]_identified_targets.fas`, `cluster_tree`, `catalytic_aas.txt`
Výstupy:	`identified_targets.fas`, `msa_identified_targets.fas` `catalytic_pentads.txt`, `catalytic_positions.txt`

Nástrojem ClustalΩ se pro každý shluk `[cutoff]_[clusterid]_identified_targets.fas` vytvoří vícenásobné zarovnání, přičemž počet zpřesňujících iterací je nastaven na 3. Na základě klíčových slov (např. *synthetic*, *vector*, apod.) obsažených v hlavičce sekvencí jsou tyto označeny za umělé a z vícenásobných zarovnání odstraněny.

Podle souboru `cluster_tree` se určí hierarchie shluků, podle níž se pak postupně provádí profile-profile MSA – začne se zarovnáním dvou nejbližších shluků, stejně se pokračuje i v dalších krocích, kdy vybraným shlukem může být dále i shluk vzniklý zarovnáním v předchozím kroku. V získaném MSA se dle informací z `catalytic_aas.txt` určí pozice katalytických reziduí a následně se u každé sekvence zkontroluje, zda obsahuje jejich kompletní sadu. Sekvence, kterým chybí na dané pozici v zarovnání katalytické reziduum (tzn. místo něj je mezera) nebo které mají na dané pozici nepovolenou aminokyselinu jsou označeny jako degenerované.

Připraví se druhá iterace tvorby MSA – z původních souborů obsahujících shluky se odstraní degenerované sekvence a znovu se spustí tvorba zarovnání, jak je popsáno výše. Opět se určí pozice katalytických reziduí a zkontroluje jejich kompletnost. Podle toho se vytvoří finální MSA `msa_identified_targets.fas` opětovným odstraněním degenerovaných sekvencí z posledního zarovnání. Vytvoří se i odpovídající finální množina sekvencí `identified_targets.fas`.

Do souborů `catalytic_positions.txt` a `catalytic_pentads.txt` se uloží pozice a typ reziduí identifikovaných sekvencí odpovídající katalytickým pozicím známých sekvencí podle výsledného zarovnání.

6.3.6 PdbBlast

Umístění:	`loschmidt.hts.module.pdbblast`
Nástroje:	MakeBLASTdb[12], PSI-BLAST[12], TemplateFinder
Vstupy:	PDB databáze, `identified_targets.fas`, `msa_identified_targets.fas`, `catalytic_aas.txt`, `catalytic_positions.txt`, `own_structures.fas`, `o_catalytic.txt`, `o_[nn].cif`, `o_[nn].pdb`
Výstupy:	`msa_identified_targets_cut.fas`, `catalytic_pentads_cut.txt`, `catalytic_positions_cut.txt`, `catalytic_positions_templates.txt`, `removed_domains.txt`, `modeller/[n]/[m]`, `modeller/templates_db`

Z množiny identifikovaných sekvencí `identified_targets.fas` se extrahují sekvence, pro které existuje záznam v PDB databázi (dle přítomnosti řetězce `|pdb|[pdbid]|` v hlavičce). K těmto se dále přidají všechny sekvence ze souboru `own_structures`, tj. sekvence pro vlastní struktury, které nejsou dostupné v PDB databázi. Z takto sestavených sekvencí se pomocí nástroje MakeBLASTdb vytvoří vlastní vyhledávací databáze pro algoritmus PSI-BLAST.

Ve finálním MSA z `msa_identified_targets.fas` se stanoví konsensuální začátek a konec sekvencí se strukturou dostupnou v PDB databázi. Například při konsensuálním prahu 50 % je začátek stanoven na pozici MSA, kde má aspoň 50 % sekvencí reziduum a nikoliv mezeru. Určený konsensuální začátek a konec se posune o toleranční konstantu směrem k N-terminálnímu, resp. C-terminálnímu konci MSA. Např. původně určený začátek a konec na pozicích 145 a 320 se při aplikace tolerance 20 posune na pozici 125 a 340.

Vícenásobné zarovnání se ořeže dle zjištěného začátku a konce a uloží do souboru `msa_identified_targets_cut.fas` – ořezané sekvence budou použity pro konstrukci homologních modelů a tím pádem pro všechny následné strukturní analýzy. Ořezání je důležité, neboť při velkých rozdílech v délce sekvencí dochází při tvorbě párového přiložení cílové sekvence a templátu k velkým chybám. Pro každou sekvenci se zaznamená počet ořezaných reziduí do souboru `removed_domains.txt`.

Pro každou sekvenci se na základě ořezaného MSA určí katalytická rezidua a jejich pozice – výstupem jsou soubory `catalytic_pentads_cut.txt` a `catalytic_positions_cut.txt`.

Každá ořezaná sekvence je použita jako vstup pro PSI-BLAST vyhledávání oproti vlastní databázi sekvencí s PDB strukturou (viz výše). Výsledky BLASTu jsou zpracovány pomocí programu TemplateFinder, jsou identifikovány potenciální templáty a rovnou jsou pro ně staženy potřebné strukturní soubory z PDB databáze.

Pro každou sekvenci a každý její potenciální templát je ve složce `modeller/[n]/[m]` připravena úloha pro MODELLER, kde `[n]` značí pořadové číslo identifikované sekvence a `[m]` pořadové číslo možného templátu pro danou sekvenci. Úloha obsahuje soubor `[gi].fas` s ořezanou sekvencí, pro níž konstruujeme homologní model a odpovídající skript pro MODELLER, který produkuje párové přiložení sekvence na konkrétní templát.

Nakonec jsou potenciální templáty nakopírovány do složky `modeller/templates_db/` a pro každý templát jsou uloženy pozice odpovídající katalytickým reziduím do souboru `catalytic_positions_templates.txt`.

Skript spouštějící úlohy MODELLERu není zatím implementován v rámci frameworku

Loschmidtových laboratoří, protože vzhledem k jeho enormní časové náročnosti je k jeho běhu potřeba speciální výpočetní infrastruktury typu MetaCentrum. Prozatímní skripty právě pro tuto infrastrukturu – `run_job.sh`, `submit_job.sh` a `run_modeller.pl` – jsou dostupné na stejném místě jako tento modul.

6.3.7 StructureAlignment

Umístění:	`loschmidt.hts.module.structurealignment`
Nástroje:	CASTp[10], jCE[48]
Vstupy:	`aln_template.pdb`, `catalytic_pentads_cut.txt`, `modeller/[n]/`
Výstupy:	`catalytic_pocket.txt`, `selected_template_info.txt`
	`tunnels/`, `cavities/[n]/`

Připraví se úlohy pro strukturní zarovnání – každý finální homologní model z `modeller/[n]/` se nakopíruje do lokální složky s úlohou `[n]` spolu se strukturou `aln_template.pdb`, na kterou má být přiložen. Zároveň se sesbírá informace o použitých templátech a uloží se do souhrnného souboru `selected_templates_info.txt`.

Pro všechny úlohy strukturního zarovnání se spustí nástroj jCE a na výsledném zarovnaném homologním modelu se provede CASTp analýza.

Na základě informací z výstupních souborů CASTp analýzy se pro každou identifikovanou kapsu spočítá celkový počet katalytických atomů, tj. atomů všech katalytických reziduí, které danou kapsu tvoří. Kapsa s nejvyšším počtem těchto atomů je označena jako katalytická a informace o ní jsou uloženy do souboru `cavities/[n]/catalytic_pocket.txt` i do souhrnného souboru `catalytic_pocket.txt`.

Nakonec je vytvořena složka `tunnels` a v ní konfigurační soubor `config.txt`, skript pro spuštění nástroje CAVER `run_caver.sh` a vytvořena složka pro vstupní PDB soubory `pdbs`. Do ní jsou nakopírovány všechny zarovnané homologní modely s upravenými čísly vybraných atomů, aby mohl být použit stejný startovní bod pro všechny modely a aby tunely mohly být analyzovány přes všechny modely najednou, což umožní jejich shlukování podle vzájemné podobnosti a zpřehlednění výsledků.

6.3.8 BioProject

Umístění:	`loschmidt.hts.module.bioproject`
Vstupy:	BioProject[34] a Taxonomy[36, 35] databáze, `identified_targets.fas`
Výstupy:	`extremophilicity.txt`

Z hlaviček sekvencí ze souboru `identified_targets.fas` se extrahují jména všech organismů – vždy celé jméno včetně kmenu a k tomu zkrácené – tj. jen rodové a druhové jméno.

Pro každý organismus se z databáze BioProject pokud možno zjistí informace o salinitě, optimální teplotě a rozmezí teplot.

Podle souboru `names.dmp`[36] z databáze Taxonomy se přiřadí každému organismu číslo taxonu. Spojením čísla taxonu a mapovacího souboru `categories.dmp`[35] se určí říše, do které organismus patří.

Nakonec se pro každou sekvenci uloží údaj o hostitelském organismu a jeho vlastnostech do `extremophilicity.txt`. Pokud je určité GI číslo asociováno s více organismy, je každý záznam uveden na zvláštním řádku.

6.3.9 Pfam

Umístění: `loschmidt.hts.module.pfam`

Vstupy: Pfam databáze[17], `identified_targets.fas`

Výstupy: `pfam_parsed.txt`

Pfam k vyhledávání domén poskytuje RESTful rozhraní – dotaz musí být formulován jako HTTP požadavek typu POST, v jehož těle je nutné specifikovat několik hodnot ve tvaru `multipart/form-data`[33], a odeslán na adresu `http://pfam.xfam.org/search/sequence`. Hodnota s klíčem `seq` musí obsahovat sekvenci, která má být použita jako dotaz do Pfam databáze a hodnota s klíčem `output` určuje výstupní formát (pro potřeby tohoto modulu nastaveno na `xml`). Pfam v reakci na dotaz vrátí XML soubor, který obsahuje URL pro vyzvednutí výsledku po doběhnutí vyhledávání.

Proces zjišťování domén probíhá tak, že se ze souboru `identified_targets.fas` extrahují jednotlivé sekvence a použijí se s časovým rozestupem několika sekund jako dotaz na RESTful rozhraní Pfam. Vrácené adresy pro vyzvednutí výsledků jsou uloženy a v druhé fázi se výsledky postupně stahují pro všechny úlohy spuštěné na Pfam serveru. Z nich se zjistí přístupový kód identifikované domény, název domény, začátek a konec domény v rámci sekvence a uloží se do souboru `pfam_parsed.txt`.

RESTful rozhraní Pfam se nicméně jeví jako nespolehlivé – stává se, že někdy nevrátí žádné výsledky, bez ohledu na dobu, po kterou se na ně čeká. Jako částečné řešení je implementován základní způsob zotavení z chyby – pokud se nepodaří získat výsledky, celý proces se pro danou sekvenci opakuje, maximálně však třikrát.

Kapitola 7

Experimentální ověření

Z popisu povinných vstupů pro vyhledávání (viz 5.1) je zřejmé, že výsledný nástroj vyžaduje poměrně velké množství expertních znalostí – ať už to jsou známé sekvence enzymů, specifikace katalytických reziduí nebo znalost struktury. Z tohoto důvodu byla funkce prezentovaného algoritmu ověřena na proteinové rodině haloalkan dehalogenáz, jejímuž inženýrství se věnuje již zmíněný tým Loschmidtových laboratoří[30], a byli proto ochotni poskytnout dostatek znalostí, zkušeností a prostředků, jak pro správnou specifikaci všech vstupů, tak i případné ověření a experimentální charakterizaci putativních proteinů z této rodiny přímo v laboratoři.

V úvodu této kapitoly je rodina haloalkan dehalogenáz stručně představena, ať už z hlediska funkce, struktury, nebo i možné aplikace v průmyslu. Následuje přehled vstupních dat pro vyhledávání a zdůvodnění jejich výběru. Nakonec jsou prezentována některá výstupní data z jednotlivých logických fází v pořadí, jak jsou uvedeny na obrázku 5.1.

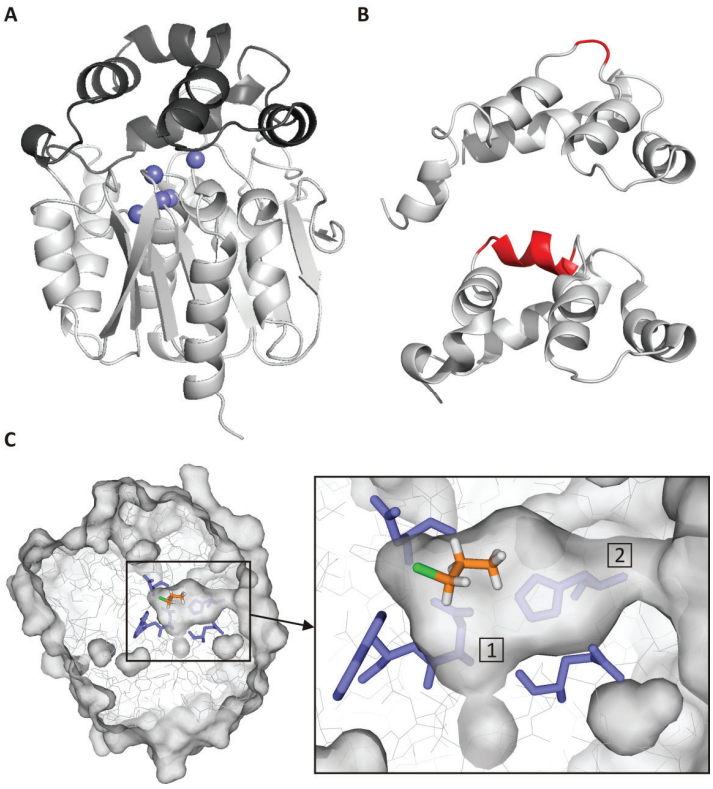
7.1 Rodina haloalkan dehalogenáz

Haloalkan dehalogenázy (EC[57] 3.8.1.5) jsou mikrobiální enzymy schopné hydrolyticky degradovat halogenované alifatické uhlovodíky na příslušný primární alkohol, halogenidový aniont a proton. Strukturně jsou řazeny do proteinové nadrodiny α/β hydroláz a vyznačují se přítomností dvou domén (hlavní a víčkové), mezi nimiž se nachází aktivní místo – viz obrázek 7.1.

Dehalogenázy umožňují rozklad řady toxických látek znečisťujících životní prostředí na produkty, které již toxické nejsou. Z tohoto důvodu mají dehalogenázy velký aplikační a inovační potenciál v oblasti biologického odbourávání toxického odpadu[52] nebo při detekci halogenovaných sloučenin v životním prostředí. Další oblasti využití dehalogenáz zahrnují biokatalýzu, detoxifikaci území zamořených bojovými plyny (např. yperitem), produkci proteinů nebo vizualizaci buněk[27]. Kromě toho se dehalogenázy jeví i jako vhodný modelový systém pro studium vztahů mezi strukturou proteinu a jeho funkcí.

7.2 Hlavní vstupy

Jak vyplývá z předchozího textu v kapitolách 5 a 6 a především z rejstříku sdílených souborů 6.2, hlavní vstupy vyhledávacího algoritmu tvoří několik souborů, které specifikují druh a funkci proteinu, který se má vyhledávat. V této podkapitole je rozvedena definice vstupů na míru upravená pro hledání příbuzných proteinů z rodiny haloalkan dehalogenáz.



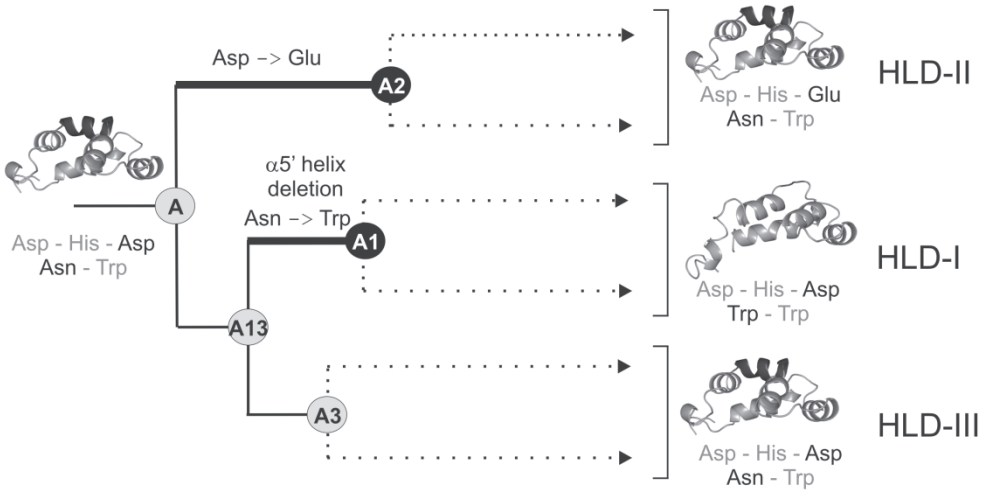
Obrázek 7.1: *Typická terciární struktura dehalogenáz*[14] – (A) hlavní, resp. víčková doména je zobrazena světle, resp. tmavě šedou barvou. Umístění reziduí katalytické pentády je ilustrováno modrými kuličkami. (B) dvě různé kompozice víčkové domény v dehalogenázách, hlavní rozdíl je zvýrazněn červenou barvou. (C) řez strukturou s detailem dutiny aktivního místa (1) a hlavního přístupového tunelu (2). Rezidua katalytické pentády, resp. molekula substrátu jsou zvýrazněny modrou, resp. oranžovou barvou.

Sekvence pro vyhledávání (`queries.fas`)

Na základě fylogenetické analýzy experimentálně potvrzených i putativních dehalogenáz z databází proteinových sekvencí byla rodina dehalogenáz klasifikována do tří základních podrodin: HLD-I, HLD-II, a HLD-III[13] (viz obrázek 7.2). Soubor `queries.fas` proto obsahuje po jednom zástupci z každé podrodiny, aby byly ve výsledcích zahrnuty příbuzné sekvence všech známých druhů dehalogenáz. Konkrétně se jedná o proteiny s označením DhlA (HLD-I), LinB (HLD-II) a DrbA (HLD-III). K těmto sekvencím je pro zvýšení různorodosti výsledků připojena i poměrně odlišná sekvence dehalogenázy z organismu *Aspergillus niger* tvořící pravděpodobně novou podrodinu HLD-IV. Celý obsah souboru `queries.fas` naleznete v příloze B.1.

Známé sekvence (`known_sequences.fas`)

Do množiny známých sekvencí bylo vybráno 23 experimentálně potvrzených dehalogenáz. Kromě čtyř sekvencí použitých jako sekvence pro vyhledávání (viz výše), i 19 dalších sekvencí s označením: DmbC, DhcA, DppA, DhmA, DmbB, DpcA, DmsA, DmbA, Luc, DpsA,



Obrázek 7.2: *Pravděpodobný průběh evoluce katalytické pentády a víčkové domény v rámci rodiny haloalkan dehalogenáz*[14] – společný předek je uveden v kořeni stromu (A). Evoluce vedla ke změně katalytické kyseliny z Asp (kyseliny asparagové) na Glu (kyselinu glutamovou) podél větve vedoucí k předku (A2) podrodiny HLD-II a ke změně halid-stabilizujícího rezidua z Asn (asparagin) na Trp (tryptofan) podél větve od společného předka podrodin HLD-I a HLD-III (A13) směrem k společnému předku podrodiny HLD-I (A1). Evoluce víčkové domény vyústila v deleci jejího druhého helixu v podrodině HLD-I.

Sav4779, DatA, DmxA, DmlA, DbjA, DbeA, DhaA, Jann2620 a DmmA. Celý obsah souboru `known_sequences.fas` naleznete v příloze B.2.

Specifikace katalytických reziduí (`catalytic_aas.txt`)

Pro rodinu haloalkan dehalogenáz je typické, že jejich katalytická funkce je podmíněna přítomností správné kombinace pěti katalytických reziduí v aktivním místě – tzv. pentády. Každá podrodina je přitom svou kompozicí katalytické pentády specifická. Tvar základní triády – tj. kyseliny asparagové jako nukleofilu, histidinu jako katalytické zásady a tryptofanu jako jednoho z halid-stabilizujících reziduí – je sice konzervovaný přes všechny podrodiny, ale jak typ, tak prostorové umístění katalytické kyseliny a druhého halid-stabilizujícího rezidua, se v různých podrodinách liší (viz obrázek 7.2). Všechny tyto kombinace jsou zachyceny ve vstupním souboru `catalytic_aas.txt`, který je textovou reprezentací tabulky 7.1 – vždy pro dvě vybrané sekvence z každé podrodiny specifikuje pozice katalytických reziduí konkrétní varianty pentády.

Struktury a jejich sekvence (`aln_template.pdb`, `own_structures/`)

Jako struktura `aln_template.pdb`, na kterou budou přiloženy všechny homologní modely byla vybrána struktura s PDB kódem 3U1T (řetězec A). Jedná se o experimentálně zjištěnou strukturu dehalogenázy s označením DmmA metodou rentgenové krystalografie a dobře reprezentuje obvyklou strukturu známých dehalogenáz.

Do složky `own_structures/` bylo přidáno několik dosud nezveřejněných struktur a odpovídajících sekvencí, se kterými Loschmidtovy laboratoře pracují. Čtyři struktury dehalo-

Sekvence	Nukleofil	Kys. 1	Kys. 2	Zásada	Hal. 1	Hal. 2	Hal. 3
DrbA	139	–	272	300	71	140	–
DmbC	109	–	238	267	43	110	–
DmbB	123	–	250	279	–	124	164
DhlA	124	–	260	289	–	125	175
LinB	108	132	–	272	38	109	–
DhaA	106	130	–	272	41	107	–

Tabulka 7.1: *Pozice katalytických reziduí pentády ve vybraných sekvencích.* Na pozici nukleofilu může být pouze Asp, na pozici kyselin Asp nebo Glu, na pozici zásady pouze His a na pozici halid-stabilizujících reziduí Asn, Gln, Trp, Tyr nebo His.

genáz z organismu *Bradyrhizobium elkanii* a dvě z *Agrobacterium fabrum*.

7.3 Hledání podobných sekvencí

Výstupem logické fáze hledání podobných sekvencí z databáze NR, toho času o velikosti 45 miliónů sekvencí, je celkem 10 251 unikátních výsledků. Zastoupení dílčích podrodin ve výsledcích není rovnoměrné, což není překvapující vzhledem k faktu, že podrodina HLD-III je obecně nejčastější a nejvíce studovaná. Pro představu, při vstupní sekvenci DrbA z podrodiny HLD-III byl počet výsledků 8 840, při DhlA z HLD-I 5 703, při LinB z HLD-II 5 684 a při dodatečné sekvenci z *Aspergillus niger* z nové podrodiny HLD-IV 487. Tabulka 7.2 pak ukazuje zastoupení dílčích podrodin v rámci finálních unikátních výsledků po odstranění duplicit (detaily a důvody odstranění viz kapitola 6.3.1).

HLD-I	HLD-II	HLD-III	HLD-IV	$\sum$
17 %	15 %	67 %	1 %	10 251

Tabulka 7.2: *Zastoupení podrodin ve výsledcích hledání.* Pokud byla některá sekvence opakovaně nalezena při hledání v rámci různých podrodin, je přiřazena k té nejbližší podle nejnižší hodnoty E-value.

7.4 Filtrace

Logická fáze filtrace výsledků zahrnuje dva hlavní kroky – shlukování (viz 6.3.4) a konstrukci vícenásobného zarovnání (viz 6.3.5). Shlukování výsledků z předchozí fáze hledání podobných sekvencí vede na tvorbu hierarchického stromu shluků, který je ořezán na finální prahové hodnotě 74 podle pravidla výběru optimálního prahu (viz 6.3.4). Na tomto prahu jsou sekvence seskupeny do celkem 78 shluků, z nichž 4 obsahují známé sekvence ze souboru

`known_sequences.fas` a postupují do fáze konstrukce MSA – jejich charakteristiku shrnuje tabulka 7.3.

Shluk	Sekvencí	Známé sekvence
20161	13	Aspergillus niger
20402	527	DhlA, DppA, DhmA, DpcA, DmbB
20427	493	DmbA, DmsA, LinB, DpsA, Luc, DhaA, DbeA, Jann2620, DmmA, DatA, Sav4779, DmxA, DbjA, DmlA
20452	497	DrbA, DhcA, DmbC
$\sum$	1 530	

Tabulka 7.3: *Přehled finálních shluků* – počet členů a známé sekvence, které obsahují.

Z celkového počtu sekvencí ve finálních shlucích vyplývá, že shlukováním bylo odstraněno 8 721 výsledků z předchozí fáze. K další redukci dochází při konstrukci vícenásobného zarovnání – nejdříve je vyloučeno 39 sekvencí považovaných za umělé a poté dalších 434 z důvodu, že byly označeny jako nekompletní nebo umělé, což znamená, že nesplňují podmínky, které jsou kladeny na typ a umístění katalytických reziduí podle tabulky 7.1). Celkově tedy po filtraci zbývá 1 057 finálních putativních sekvencí dehalogenáz. Zajímavé může být i srovnání zastoupení jednotlivých podrodin před, resp. po filtraci – viz tabulky 7.2, resp. 7.4.

HLD-I	HLD-II	HLD-III	HLD-IV	$\sum$
27 %	35 %	37 %	1 %	1 057

Tabulka 7.4: *Zastoupení podrodin ve výsledcích po filtraci* – srov. s tabulkou 7.2.

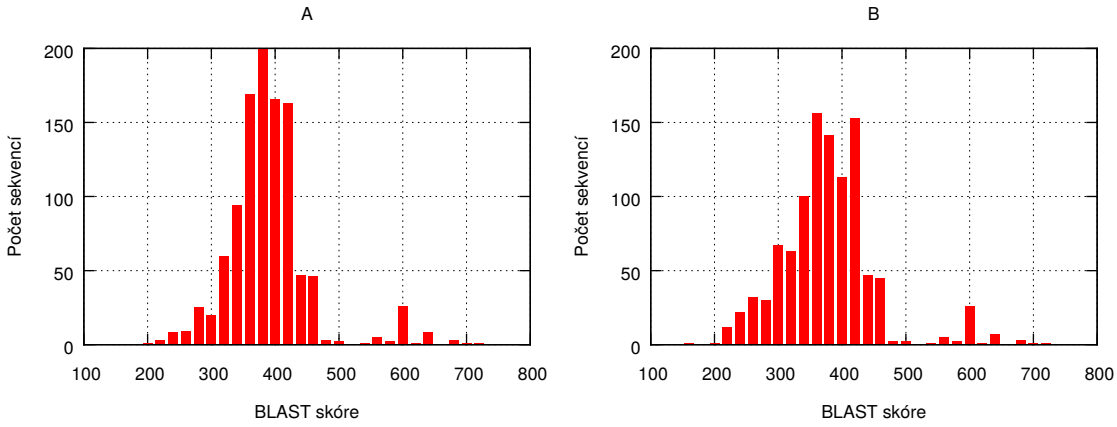
Přínos filtrace – různorodost výsledků

Poměrně komplexní algoritmus filtrace může vzbuzovat otázku, nakolik je ve skutečnosti přínosný oproti jednodušším možnostem filtrace. Pokud by podobného účinku bylo možné dosáhnout např. jednoduchým zpřísněním parametrů homologního vyhledávání, tj. snížením prahu E-value, postrádala by výpočetně a především znalostně náročná filtrace hlubšího smyslu.

Z toho důvodu byl učiněn následující srovnávací experiment: bylo provedeno modifikované homologní vyhledávání s E-value nastavenou o několik řádů níže (10^{-70} oproti původní hodnotě 10^{-20}) tak, aby počet nalezených sekvencí po triviálním odfiltrování umělých sekvencí byl přibližně roven výše zmíněné hodnotě 1 057 (viz tabulka 7.5). Porovnáním výsledků z tohoto experimentu s navrženým postupem vyšlo najevo, že při srovnatelném počtu výsledků algoritmus filtrace umožňuje získat z databáze výsledky s podstatně vyšší variabilitou, než kdybychom použili pouze užší homologní vyhledávání – výsledky po filtraci obsahují více než 20 % jiných putativních sekvencí s nižší mírou podobnosti (tzn. nižším skóre), než výsledky z jednoduššího experimentu. Tento fakt ilustruje graf 7.3.

E-value	10^{-20}	10^{-30}	10^{-40}	10^{-50}	10^{-60}	10^{-70}
Celkem	10 251	2 151	1 494	1 384	1 188	1 087
Umělé	51	39	39	39	39	39

Tabulka 7.5: *Vliv E-value na počet výsledků homologního vyhledávání* – ke každému nastavení prahu E-value je uveden i počet triviálně odfiltrovaných umělých sekvencí, které by mohly srovnání zkreslovat.



Obrázek 7.3: *Přínos filtrace* – (A) histogram rozložení skóre výsledků z experimentu s prahovou E-value nastavenou na 10^{-70}, (B) histogram rozložení skóre putativních sekvencí zbylých po filtraci. Velikost obou množin sekvencí je srovnatelně velká. U histogramu (B) je patrné vychýlení směrem k nižším hodnotám skóre a tedy k nižší míře podobnosti – tj. větší variabilitě výsledků.

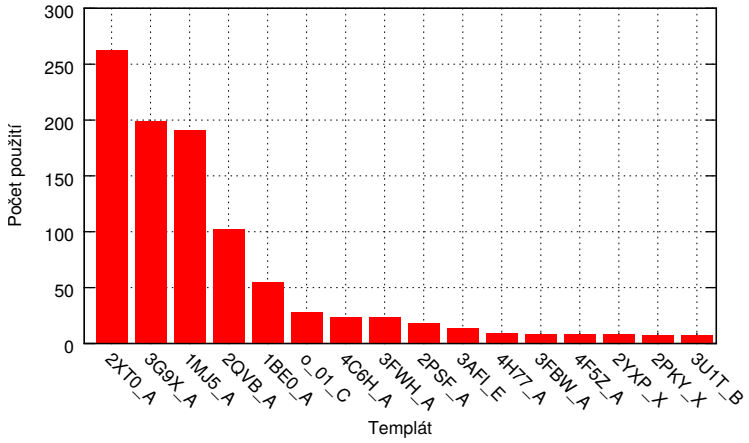
7.5 In silico screening výsledků

Poslední logickou fázi výpočtu algoritmu je možné označit jako počítačovou (in silico) charakterizaci putativních sekvencí, jejímž cílem je simulovat nákladné experimentální ověření vlastností proteinu v laboratoři.

Prvním krokem charakterizace dehalogenáz je konstrukce homologních modelů. V této fázi se podařilo vytvořit 3D model pro každou putativní dehalogenázu. Nejčastěji používaný templát pro modelování byl 2XT0 (viz graf 7.4).

Druhým krokem je hledání katalytické dutiny, tj. dutiny aktivního místa. Celkové rozložení objemu nalezených dutin napříč modely vystihuje histogram 7.5. Tunely vedoucí z aktivního místa směrem na povrch dehalogenáz jsou v třetím kroku analyzovány v jednom běhu přes všechny homologní modely, díky čemuž jsou pro nejčastější tvary vytvořeny shluky. V tabulce 7.6 je pak uvedeno několik nejčastějších shluků tunelů s několika hlavními parametry.

V posledním kroku počítačové charakterizace je pro putativní dehalogenázy pokud možno zjištěno, z jakého hostitelského organismu pochází (viz tabulka 7.7) a z dostupných experimentálních dat jsou extrahovány záznamy o předpokladech pro funkci za extrémních podmínek (viz tabulka 7.8).



Obrázek 7.4: *Nejčastěji používané templáty* – 15 nejčastějších templátů využitých při homologním modelování. Na šestém místě se umístila vlastní struktura s číslem `01` ze složky `own_structures/`.

Shluk	Počet	Prům. BR [Å]	Max. BR [Å]	Prům. délka [Å]
1	1 010	1,80	3,12	12.22
2	829	1,29	2,28	16,00
3	593	1,27	2,31	15,30
4	369	1,13	1,85	23,27
5	321	1,17	2,28	26,21

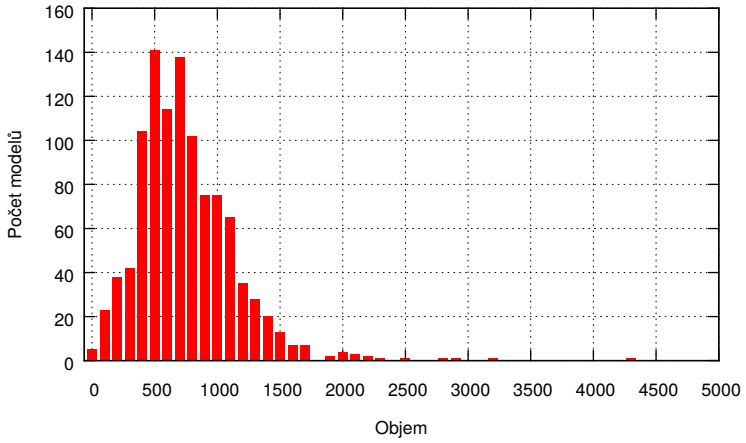
Tabulka 7.6: *Nejpočetnější shluky tunelů a jejich parametry.* BR – poloměr úzkého hrdla tunelu, počet – množství tunelů přiřazených do daného shluku.

Archea	Bacteria	Eukaryota	$\sum$
5	939	35	969

Tabulka 7.7: *Zastoupení říší hostitelských organismů* – celkový součet není roven celkovému počtu putativních sekvencí (1 057), protože ne pro každou sekvenci je možné nalézt záznam v databázi Taxonomy.

Termofilní	Psychrofilní	Mírně halofilní	Extrémně halofilní
3	24	24	5

Tabulka 7.8: *Počty organismů s extrémními vlastnostmi* – informace z databáze BioProject jsou dostupné jen pro menší část putativních sekvencí (zhruba 400), z nichž většina je označena jako mezofilní (tj. běžné).



Obrázek 7.5: *Rozložení objemu nalezených dutin napříč modely* – nejčastěji se vyskytují dutiny o objemu okolo 700 $Å^3$ (objem odvozený od molekulárním povrchu), výjimečně u některých homologních modelů jsou dutiny pětkrát i šestkrát větší.

7.6 Poznámky k výpočetní náročnosti

Prezentovaný algoritmus pro vyhledávání příbuzných proteinů s modifikovanou funkcí je poměrně vysoce výpočetně náročný. Pro uvedenou případovou studii na rodině haloalkan dehalogenáz se čas výpočtu pohybuje okolo 600 CPU hodin. Nejvíce však k celkovému času přispívá fáze tvorby homologních modelů, která v případě konstrukce 1 057 modelů dehalogenáz spotřebovala přibližně 550 CPU hodin a bylo proto vhodné tento krok provádět na speciální výpočetní architektuře MetaCentra. Pokud by měla být provedena analýza rodiny větších proteinů, než jsou středně velké dehalogenázy, mohla by se časová náročnost i řádově zvýšit.

Kapitola 8

Závěr

Cílem tohoto diplomového projektu bylo seznámit se se základními principy proteinového inženýrství a způsoby získávání proteinů s požadovanými vlastnostmi, dále prostudovat existujícími přístupy a nástroje pro identifikaci příbuzných proteinů s modifikovanou funkcí a především pak navrhnout a implementovat nový nástroj pro vyhledávání příbuzných proteinů s modifikovanou funkcí. Při návrhu měla být zohledněna struktura proteinu, pozice aktivního místa, umístění tunelů, kapes apod.

Základní principy proteinového inženýrství dokumentuje kapitola 3, existující přístupy a nástroje jsou pak popsány v kapitole 4. Bylo zjištěno, že doposud nikdo nenavrhl takový nástroj, který by automatizovaně umožňoval vyhledávání příbuzných proteinů s modifikovanou funkcí v biologických databázích. Existující přístupy a nástroje řeší tento problém vždy jen částečně. První skupina nástrojů se například zabývá pouze hledáním příbuzných sekvencí a funkčnost proteinu dále neověřuje. Druhá skupina přístupů se pak snaží na základě sekvence nebo i strukturních informací odvodit funkci proteinu.

Proto byl navržen a implementován nový nástroj, který obě skupiny nástrojů a přístupů spojuje v komplexnější analýzu, která u nalezených příbuzných proteinů navíc ověřuje i zachování jejich funkce. Uchování funkce je definováno přítomností všech nutných katalytických reziduí v putativní sekvenci, správnou pozicí katalytických reziduí v aktivním místě, dostatečnou velikostí přístupových tunelů k aktivnímu místu a dalšími vlastnostmi nalezených sekvencí (např. nezajímají nás sekvence anotované jako umělé, apod.). Klíčovým krokem analýzy je vytvoření homologního strukturního modelu pro každou putativní sekvenci se zafixovanou pozicí katalytických reziduí oproti známé referenční struktuře zadané uživatelem. Pokud se takto podaří vytvořit kvalitní homologní model, je to dobrý předpoklad pro to, aby měl i putativní protein funkci stejného typu.

Funkce nástroje byla ověřena na proteinové rodině haloalkan dehalogenáz ve spolupráci s Loschmidtovými laboratořemi a ukázalo se, že poskytuje dostatečně různorodé a kvalitně anotované výsledky, které jsou zajímavé pro proteinové inženýry a stojí za to vyzkoušet experimentální charakterizaci některých z nich přímo v laboratoři.

Do budoucna zůstává v každé fázi navrženého nástroje jistý prostor pro další vylepšení. Ať už se jedná o fázi hledání homologních sekvencí, filtraci podle specifikace katalytických reziduí nebo počítačovou charakterizaci výsledků. Veškeré další práce by pak mohly vyústit v implementaci uživatelského rozhraní (např. webového), které by nástroj zpřístupnilo širší vědecké obci formou výpočetní služby. S přihlédnutím k enormní časové náročnosti celého algoritmu se ale nabízí i otázka, jak ji snížit, aby provoz takové služby byl technicky i finančně zvládnutelný. Jedno z řešení by mohlo implementovat inkrementální způsob výpočtu, tzn. při opakovaném spuštění nástroje se stejnými parametry, ale novou verzí databáze, by se

prováděla analýza jen nově přidaných nebo aktualizovaných sekvencí. Interakce s uživatelem by mohla vypadat následovně: uživatel jednou specifikuje všechny parametry vyhledávání a zvolí interval, ve kterém chce analýzu opakovaně provádět. V daných intervalech poté obdrží pouze shrnutí nových identifikovaných sekvencí, které se v poslední době objevily, což mu umožní být neustále v kontaktu s výsledky aktuálních výzkumů a sekvenačních projektů.

Literatura

[1] Akoh, C. C.; Chang, S. W.; Lee, G. C.; aj.: Biocatalysis for the production of industrial products and functional foods from rice and other agricultural produce. *J. Agric. Food Chem.*, ročník 56, č. 22, 2008: s. 10445–10451.

[2] Alberts, B.; Bray, D.; Johnson, A.; aj.: *Základy buněčné biologie – Úvod do molekulární biologie buňky*. Espero Publishing, druhé vydání, 2005, ISBN 80-902906-2-0.

[3] Altschul, S. F.; Gish, W.; Miller, W.; aj.: Basic local alignment search tool. *Journal of Molecular Biology*, ročník 215, č. 3, 1990: s. 403–410.
URL http://linkinghub.elsevier.com/retrieve/pii/S0022283605803602

[4] Ayala, M.; Pickard, M. A.; Vazquez-Duhalt, R.: Fungal enzymes for environmental purposes, a molecular biology challenge. *J. Mol. Microbiol. Biotechnol.*, ročník 15, č. 2-3, 2008: s. 172–180.

[5] Bairoch, A.; Boeckmann, B.: The SWISS-PROT protein sequence data bank: current status. *Nucleic Acids Res.*, ročník 22, č. 17, 1994: s. 3578–3580.

[6] Banta, S.; Wheeldon, I. R.; Blenner, M.: Protein engineering in the development of functional hydrogels. *Annu Rev Biomed Eng*, ročník 12, 2010: s. 167–186.

[7] Barker, W. C.; George, D. G.; Hunt, L. T.; aj.: The PIR protein sequence database. *Nucleic Acids Res.*, ročník 19 Suppl, 1991: s. 2231–2236.

[8] Bendl, J.; Stourac, J.; Salanda, O.; aj.: PredictSNP: Robust and Accurate Consensus Classifier for Prediction of Disease-Related Mutations. *PLoS Computational Biology*, ročník 10, č. 1, 2014.
URL http://dx.plos.org/10.1371/journal.pcbi.1003440

[9] Berman, H. M.; Westbrook, J.; Feng, Z.; aj.: The Protein Data Bank. *Nucleic Acids Res.*, ročník 28, č. 1, 2000: s. 235–242.

[10] Binkowski, T. A.; Naghibzadeh, S.; Liang, J.: CASTp: Computed Atlas of Surface Topography of proteins. *Nucleic Acids Res.*, ročník 31, č. 13, 2003: s. 3352–3355.

[11] Borgne, S. L.; Quintero, R.: Biotechnological processes for the refining of petroleum. *Fuel Processing Technology*, ročník 81, č. 2, 2003: s. 155–169.
URL http://linkinghub.elsevier.com/retrieve/pii/S0378382003000079

[12] Camacho, C.; Coulouris, G.; Avagyan, V.; aj.: BLAST+: architecture and applications. *BMC Bioinformatics*, ročník 10, 2009: str. 421.

[13] Chovancova, E.; Kosinski, J.; Bujnicki, J. M.; aj.: Phylogenetic analysis of haloalkane dehalogenases. *Proteins*, ročník 67, č. 2, 2007: s. 305–316.

[14] Chovancová, E.: *Bioinformatická analýza a design halogenalkandehalogenas.* Disertační práce, Masarykova univerzita, Přírodovědecká fakulta, 2011.
URL http://is.muni.cz/th/40707/prif_d/

[15] Edgar, R. C.: Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, ročník 26, č. 19, 2010: s. 2460–2461.

[16] Fersht, A.: *Enzyme structure and mechanism.* New York: W.H. Freeman, druhé vydání, 1985.

[17] Finn, R. D.; Mistry, J.; Tate, J.; aj.: The Pfam protein families database. *Nucleic Acids Res.*, ročník 38, č. Database issue, 2010: s. D211–222.

[18] Fischer, E.: Einfluss der Configuration auf die Wirkung der Enzyme. *Berichte der deutschen chemischen Gesellschaft*, ročník 27, č. 3, 1894: s. 2985–2993.
URL http://doi.wiley.com/10.1002/cber.18940270364

[19] Fiser, A.; Sali, A.: Modeller: generation and refinement of homology-based protein structure models. *Meth. Enzymol.*, ročník 374, 2003: s. 461–491.

[20] Geourjon, C.; Combet, C.; Blanchet, C.; aj.: Identification of related proteins with weak sequence identity using secondary structure information. *Protein Sci.*, ročník 10, č. 4, 2001: s. 788–797.

[21] Gupta, R.; Beg, Q. K.; Lorenz, P.: Bacterial alkaline proteases: molecular approaches and industrial applications. *Appl. Microbiol. Biotechnol.*, ročník 59, č. 1, 2002: s. 15–32.

[22] Henikoff, S.; Henikoff, J. G.: Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.*, ročník 89, č. 22, 1992: s. 10915–10919.

[23] James, J.; Simpson, B. K.: Application of enzymes in food processing. *Crit Rev Food Sci Nutr*, ročník 36, č. 5, 1996: s. 437–463.

[24] Katsuki, S.; Murad, F.: Regulation of adenosine cyclic 3',5'-monophosphate and guanosine cyclic 3',5'-monophosphate levels and contractility in bovine tracheal smooth muscle. *Mol. Pharmacol.*, ročník 13, č. 2, 1977: s. 330–341.

[25] Kirk, O.; Borchert, T. V.; Fuglsang, C. C.: Industrial enzyme applications. *Curr. Opin. Biotechnol.*, ročník 13, č. 4, 2002: s. 345–351.

[26] Koshland, D. E.: Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proceedings of the National Academy of Sciences*, ročník 44, č. 2, 1958: s. 98–104.
URL http://www.pnas.org/cgi/doi/10.1073/pnas.44.2.98

[27] Koudelakova, T.; Bidmanova, S.; Dvorak, P.; aj.: Haloalkane dehalogenases: biotechnological applications. *Biotechnol J*, ročník 8, č. 1, 2013: s. 32–45.

[28] Kozlikova, B.; Sebestova, E.; Sustr, V.; aj.: CAVER Analyst 1.0. *Bioinformatics*, ročník 30, č. 18, 2014: s. 2684–2685.
URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btu364

[29] Loewenstein, Y.; Portugaly, E.; Fromer, M.; aj.: Efficient algorithms for accurate hierarchical clustering of huge datasets. *Bioinformatics*, ročník 24, č. 13, 2008: s. i41–i49.
URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btn174

[30] Loschmidt Laboratories: Protein Engineering Group. 2015.
URL http://loschmidt.chemi.muni.cz/peg/

[31] Lutz, S.: Beyond directed evolution—semi-rational protein engineering and design. *Current Opinion in Biotechnology*, ročník 21, č. 6, 2010: s. 734–743.
URL http://linkinghub.elsevier.com/retrieve/pii/S0958166910001540

[32] Ma, B.; Tromp, J.; Li, M.: PatternHunter: faster and more sensitive homology search. *Bioinformatics*, ročník 18, č. 3, 2002: s. 440–445.
URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/18.3.440

[33] Masinter, L.: Returning Values from Forms: multipart/form-data.
URL https://www.ietf.org/rfc/rfc2388.txt

[34] NCBI: BioProject.
URL ftp://ftp.ncbi.nlm.nih.gov/bioproject/README

[35] NCBI: Taxonomy category dump.
URL ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxcat_readme.txt

[36] NCBI: Taxonomy database dump.
URL ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump_readme.txt

[37] Needleman, S. B.; Wunsch, C. D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, ročník 48, č. 3, 1970: s. 443–453.

[38] Olafsen, T.; Wu, A. M.: Antibody vectors for imaging. *Semin Nucl Med*, ročník 40, č. 3, 2010: s. 167–181.

[39] Pavelka, A.; Chovancova, E.; Damborsky, J.: HotSpot Wizard: a web server for identification of hot spots in protein engineering. *Nucleic Acids Res.*, ročník 37, č. Web Server issue, 2009: s. W376–383.

[40] Postigo, M. P.; Krogh, R.; Terni, M. F.; aj.: Enzyme kinetics, structural analysis and molecular modeling studies on a series of Schistosoma mansoni PNP inhibitors. *Journal of the Brazilian Chemical Society*, ročník 22, č. 3, 2011: s. 583–591.
URL http://www.scielo.br/scielo.php?script=sci_arttext

[41] Prokop, Z.; Damborský, J.: Comparison of rational design and directed evolution strategies. 2007.

[42] Pruitt, K. D.; Maglott, D. R.: RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Res.*, ročník 29, č. 1, 2001: s. 137–140.

[43] Reeck, G. R.; de Haen, C.; Teller, D. C.; aj.: "Homology"in proteins and nucleic acids: a terminology muddle and a way out of it. *Cell*, ročník 50, č. 5, 1987: str. 667.

[44] Rehm, B. H. A.: Bacterial polymers. *Nature Reviews Microbiology*, ročník 8, č. 8, 2010: s. 578–592.
URL http://www.nature.com/doifinder/10.1038/nrmicro2354

[45] Remmert, M.; Biegert, A.; Hauser, A.; aj.: HHblits. *Nature Methods*, ročník 9, č. 2, 2011: s. 173–175.
URL http://www.nature.com/doifinder/10.1038/nmeth.1818

[46] Sarikaya, M.; Tamerler, C.; Jen, A. K. Y.; aj.: Molecular biomimetics. *Nature Materials*, ročník 2, č. 9, 2003: s. 577–585.
URL http://www.nature.com/doifinder/10.1038/nmat964

[47] Sarker, A.; Nasreen, M.; Islam, M. R.; aj.: Computational Approach to Search for Plant Homologues of Human Heat Shock Protein. *Journal of Computer Science & Systems Biology*, ročník 6, č. 3, 2013: s. 099–105, doi:10.4172/jcsb.1000106.

[48] Shindyalov, I. N.; Bourne, P. E.: Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng.*, ročník 11, č. 9, 1998: s. 739–747.

[49] Sievers, F.; Wilm, A.; Dineen, D.; aj.: Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, ročník 7, č. 1, 2011: s. 539–539.
URL http://msb.embopress.org/cgi/doi/10.1038/msb.2011.75

[50] Sigrist, C. J.; Cerutti, L.; Hulo, N.; aj.: PROSITE: a documented database using patterns and profiles as motif descriptors. *Brief. Bioinformatics*, ročník 3, č. 3, 2002: s. 265–274.

[51] Smith, T. F.; Waterman, M. S.: Identification of common molecular subsequences. *J. Mol. Biol.*, ročník 147, č. 1, 1981: s. 195–197.

[52] Stucki, G.; Thueer, M.: Experiences of a large-scale application of 1,2-dichloroethane degrading microorganisms for groundwater treatment. *Environ. Sci. Technol.*, ročník 29, č. 9, 1995: s. 2339–2345.

[53] Tamerler, C.; Khatayevich, D.; Gungormus, M.; aj.: Molecular biomimetics: GEPI-based biological routes to technology. *Biopolymers*, ročník 94, č. 1, 2010: s. 78–94.

[54] Turanli-Yildiz, B.; Alkim, C.; Petek, Z.: Protein Engineering Methods and Applications. *Protein Engineering*, 2012.
URL http://www.intechopen.com/books/protein-engineering/protein-engineering-methods-and-applications

[55] Ulmer, K.: Protein engineering. *Science*, ročník 219, č. 4585, 1983: s. 666–671.
URL http://www.sciencemag.org/cgi/doi/10.1126/science.6572017

[56] Vickers, T.: Wikipedia. 2001.
URL https://commons.wikimedia.org/wiki/File:Induced_fit_diagram.svg

[57] Webb, E. C.: *Enzyme nomenclature 1992*. San Diego: Published for the International Union of Biochemistry and Molecular Biology by Academic Press, 1992, ISBN 01-222-7165-3.

[58] Wikipedia: Screening. 2015.
URL http://cs.wikipedia.org/wiki/Screening

[59] Zafir-Lavie, I.; Michaeli, Y.; Reiter, Y.: Novel antibodies as anticancer agents. *Oncogene*, ročník 26, č. 25, 2007: s. 3714–3733.
URL http://www.nature.com/doifinder/10.1038/sj.onc.1210372

Příloha A

Příkad konfiguračního souboru

Formát odpovídá neformálnímu standardu konfiguračních souborů INI. Sekce `[command]` je povinná a musí specifikovat minimálně hodnotu proměnné `cmd`, tj. absolutní cestu ke spustitelnému souboru nástroje. Sekce `[params-*]` definují parametry příkazové řádky pro různé případy použití nástroje, např. v rámci modulu HomologySearch 6.3.1 nebo PdbBlast 6.3.6.

```
[command]
cmd=/environment/software/ncbi-blast/bin/psiblast
dbpath=/data/blast/

; Default command params
[params-default]
-db=nr
-evalue=1e-3
-num_descriptions=100
-num_alignments=0
-num_iterations=1
-inclusion_ethresh=1e-3

[params-homsearch]
-db=nr
-evalue=1e-20
-inclusion_ethresh=1e-20
-num_iterations=2
-num_alignments=0
-num_descriptions=20000

[params-pdbblast]
-evalue=1e-3
-inclusion_ethresh=1e-3
-num_iterations=1
-max_target_seqs=5000
-outfmt=5
```

Příloha B

Příklady vstupních souborů

B.1 queries.fas

```
>Rhobal-32476333(DrbA)
MSCRLSSNRRGSSKLAAMTNLASDLFPHPSSELSIDGHTLRYIDTAASSDIPSSAVGSSD
GEPTFLCVHGNPTWSFYYRRIIERYGKQQRVIAVDHIGCGRSDKPSEDEFPYTMAAHRDN
LIRLVDELDLKNVILIAHDWGGAIGLSAMHARRDRLAGIGLLNTAAFPPPYMPQRIAACR
MPVLGTPAVRGLNLFARAAVTMAMSRTKMKPDVAAGLLAPYDNWKNRVAIDRFVRDIPLN
DSHPTMKTLRQLESDLPDLASLPISLIWGMKDWCFRPECLRRFQSVWPDAEVTELATTGH
YVIEDSPEETLAAIDSLLARVKERIGAA

>Xanaut-442872(DhlA)
MINAIRTPDQRFSNLDQYPFSPNYLDDLPGYPGLRAHYLDEGNSDAEDVFLCLHGEPTWS
YLYRKMIPVFAESGARVIAPDFFGFGKSDKPVDEEDYTFEFHRNFLLALIERLDLRNITL
VVQDWGGFLGLTLPMADPSRFKRLIIMNACLMTDPVTQPAFSAFVTQPADGFTAWKYDLV
TPSDLRLDQFMKRWAPTLTEAEASAYAAPFPDTSYQAGVRKFPKMVAQRDQACIDISTEA
ISFWQNDWNGQTFMAIGMKDKLLGPDVMYPMKALINGCPEPLEIADAGHFVQEFGEQVAR
EALKHFAETE

>Sphjap-4521186(LinB)
MSLGAKPFGEKKFIEIKGRRMAYIDEGTGDPILFQHGNPTSSYLWRNIMPHCAGLGRLIA
CDLIGMGDSDKLDPSGPERYAYAEHRDYLDALWEALDLGDRVVLVVHDWGSALGFDWARR
HRERVQGIAYMEAIAMPIEWADFPEQDRDLFQAFRSQAGEELVLQDNVFVEQVLPGLILR
PLSEAEMAAYREPFLAAGEARRPTLSWPRQIPIAGTPADVVAIARDYAGWLSESPIPKLF
INAEPGALTTGRMRDFCRTWPNQTEITVAGAHFIQEDSPDEIGAAIAAFVRRLRPA

>Aspnig-350639731(90-432)
VRRQGPGWMPDIPWADCTEEMAICGATVRFVHQKPTLDGPADRRRPIIFLHGNPSWSYIW
RNILPHLVDQGHEVYALDWLGHGRSDKATHPESVTFELHIHTLLQLFKVHNLRHAAIVAH
DWGGCVALCAIPQLPQMACTRLLLLNSFLPPRPDEWNLHYGLLYAIWFLSTGLLGGYVPE
SGVMRFMSPHLSQPEVDGYSAPYQSLPVATKASVFRFGHIVPVTPRFALHGLRQTRLWKL
LEGLCGPRHFDNLSQQARLAKRGEEVRQSWRAGLEKGNPDVAVVFGEQDPLLKLYKDILV
DCIHPAHMVEWATQGIWLSGGGHYPMEEKAADISLLAEKLASQ
```

B.2 known_sequences.fas

```
>DrbA
MSCRLSSNRRGSSKLAAMTNLASDLFPHPSSELSIDGHTLRYIDTAASSDIPSSAVGSSD
GEPTFLCVHGNPTWSFYYRRIIERYGKQQRVIAVDHIGCGRSDKPSEDEFPYTMAAHRDN
LIRLVDELDLKNVILIAHDWGGAIGLSAMHARRDRLAGIGLLNTAAFPPPYMPQRIAACR
MPVLGTPAVRGLNLFARAAVTMAMSRTKMKPDVAAGLLAPYDNWKNRVAIDRFVRDIPLN
DSHPTMKTLRQLESDLPDLASLPISLIWGMKDWCFRPECLRRFQSVWPDAEVTELATTGH
YVIEDSPEETLAAIDSLLARVKERIGAA
>DmbC
MSIDFTPDPQLYPFESRWFDSSRGRIHYVDEGTGPPILLCHGNPTWSFLYRDIIVALRDR
FRCVAPDYLGFGLSERPSGFGYQIDEHARVIGEFVDHLGLDRYLSMGQDWGGPISMAVAV
ERADRVRGVVLGNTWFWPADTLAMKAFSRVMSSPPVQYAILRRNFFVERLIPAGTEHRPS
SAVMAHYRAVQPNAAARRGVAEMPKQILAARPLLARLAREVPATLGTKPTLLIWGMKDVA
FRPKTIIPRLSATFPDHVLVELPNAKHFIQEDAPDRIAAAIIERFG
>DhcA
MDDQWINREEYPFASHYIELGPGRMHYVDEGQGKPIVMLHGNPTWSFLYRNMIKGLSGQF
RCIAPDHLGFGLSDKPGDWSYLPQDHARNLEILIKKLNLQDITLVVQDWGGPIGLSYALN
HPQNVRRLVIMNTWMWSAHGDVKYETFSTLMGGLFGRFMIKRFNVFVSVLLKQAVQKKLS
ADIYRHYSEPLKNPAERKGCWVFPKQIIKADEWLEELWEKRANIADIPTLLLWGMKDFAF
QERELRTWMQLLNHHTAVTFDDAGHFLPEDKGVEVCPLIADFLQDAEPTFQREASAPAPV
V
>DppA
MEFVRTPDDRFADLPDFPYAPHYLEGLPGFEGLRMHYVDEGPRDAEHTFLCLHGEPSWSF
LYRKMLPVFTAAGGRVVAPDLFGFGRSDKPTDDAVYTFGFHRRSLLAFLDALQLERVTLV
CQDWGGILGLTLPVDRPQLVDRLIVMNTALAVGLSPGKGFESWRDFVANSPDLDVGKLMQ
RAIPGITDAEVAAYDAPFPGPEFKAGVRRFPAIVPITPDMEGAEIGRQAMSFWSTQWSGP
TFMAVGPQDPVLGPEVMGMLRQAIGGCPEPMIVEAGGHFVQEHGEPIARAALAAFGQ
>DhlA
MINAIRTPDQRFSNLDQYPFSPNYLDDLPGYPGLRAHYLDEGNSDAEDVFLCLHGEPTWS
YLYRKMIPVFAESGARVIAPDFFGFGKSDKPVDEEDYTFEFHRNFLLALIERLDLRNITL
VVQDWGGFLGLTLPMADPSRFKRLIIMNACLMTDPVTQPAFSAFVTQPADGFTAWKYDLV
TPSDLRLDQFMKRWAPTLTEAEASAYAAPFPDTSYQAGVRKFPKMVAQRDQACIDISTEA
ISFWQNDWNGQTFMAIGMKDKLLGPDVMYPMKALINGCPEPLEIADAGHFVQEFGEQVAR
EALKHFAETE
>DhmA
MHVLRTPDSRFENLEDYPFVAHYLDVTARDTRPLRMHYLDEGPIDGPPIVLLHGEPTWSY
LYRTMITPLTDAGNRVLAPDLIGFGRSDKPSRIEDYSYQRHVDWVVSWFEHLNLSDVTLF
VQDWGSLIGLRIAAEQPDRVGRLVVANGFLPTAQRRTPPAFYAWRAFARYSPVLPAGRIV
SVGTVRRVSSKVRAGYDAPFPDKTYQAGARAFPQLVPTSPADPAIPANRKAWEALGRWEK
PFLAIFGARDPILGHADSPLIKHIPGAAGQPHARINASHFIQEDRGPELAERILSWQQAL
L
>DmbB
MDVLRTPDSRFEHLVGYPFAPHYVDVTAGDTQPLRMHYVDEGPGDGPPIVLLHGEPTWSY
LYRTMIPPLSAAGHRVLAPDLIGFGRSDKPTRIEDYTYLRHVEWVTSWFENLDLHDVTLF
VQDWGSLIGLRIAAEHGDRIARLVVANGFLPAAQGRTPLPFYVWRAFARYSPVLPAGRLV
NFGTVHRVPAGVRAGYDAPFPDKTYQAGARAFPRLVPTSPDDPAVPANRAAWEALGRWDK
PFLAIFGYRDPILGQADGPLIKHIPGAAGQPHARIKASHFIQEDSGTELAERMLSWQQAT
>DpcA
MKILRTPDSRFANLPDYNFDPHYLMVDDSEDSELRVHYLDEGPRDADPVLLLHGEPSWCY
LYRKMIPILTAAGHRVIAPDLPGFGRSDKPASRTDYTYQRHVNWMQSVLDQLDLNNITLF
CQDWGGLIGLRLVAENPDRFARVAAGNTMLPTGDHDLGEGFRKWQQFSQEIPQFHVGGTI
KSGTVTKLSQAVIDAYNAPFPDESYKEGARQFPLLVPSTPDDPASENNRAAWIELSKWTK
PFITLFSDSDPVTAGGDRIMQKIIPGTKGQAHTTIANGGHFLQEDQGEKVAKLLVQFIHD
```

NPR
>DmsA
MPGSEPYGRLQYREINGKRMAYIDEARGDAIVFQHGNPSSSYLWRNVLPHTEGLGRLVAC
DLIGMGASDKLDGSGPDSYHYHENRDYLFALWDALDLGDRVTLVLHDWGGALGFDWANRH
RDRVAGIVHMETVSVPMEWDDFPDEVAQMFRGLRSPQGEEMVLENNAFIEGVLPSIVMRT
LSEEEMIHYRRPFLNAGEDRRPTLSWPRDVPLAGEPAEVVAVIEDFGEWLATSDIPKLFI
RADPGVIQGKQRILDIVRSWPNQTEITVPGTHFLQEDSADQIGEAIASFVREIRAGDNLH
REAAPEMNSAS
>DmbA
MTAFGVEPYGQPKYLEIAGKRMAYIDEGKGDAIVFQHGNPTSSYLWRNIMPHLEGLGRLV
ACDLIGMGASDKLSPSGPDRYSYGEQRDFLFALWDALDLGDHVVLVLHDWGSALGFDWAN
QHRDRVQGIAFMEAIVTPMTWADWPPAVRGVFQGFRSPQGEPMALEHNIFVERVLPGAIL
RQLSDEEMNHYRRPFVNGGEDRRPTLSWPRNLPIDGEPAEVVALVNEYRSWLEETDMPKL
FINAEPGAIITGRIRDYVRSWPNQTEITVPGVHFVQEDSPEEIGAAIAQFVRRLRSAAGV
>LinB
MSLGAKPFGEKKFIEIKGRRMAYIDEGTGDPILFQHGNPTSSYLWRNIMPHCAGLGRLIA
CDLIGMGDSDKLDPSGPERYAYAEHRDYLDALWEALDLGDRVVLVVHDWGSALGFDWARR
HRERVQGIAYMEAIAMPIEWADFPEQDRDLFQAFRSQAGEELVLQDNVFVEQVLPGLILR
PLSEAEMAAYREPFLAAGEARRPTLSWPRQIPIAGTPADVVAIARDYAGWLSESPIPKLF
INAEPGALTTGRMRDFCRTWPNQTEITVAGAHFIQEDSPDEIGAAIAAFVRRLRPA
>Luc
MTSKVYDPELRKRMITGPQWWARCKQMNVLDSFINYYDSEKHAENAVIFLHGNAASSYLW
RHVVPHVEPVARCIIPDLIGMGKSGKSGNGSYRLLDHYKYLTEWFKHLNLPKKIIFVGHD
WGACLAFHYCYEHQDRIKAVVHAESVVDVIESWDEWPDIEEDIALIKSEEGEKMVLENNF
FVETMLPSKIMRKLEPEEFAAYLEPFKEKGEVRRPTLSWPREIPLVKGGKPDVVEIVRNY
NAYLRASHDLPKMFIESDPGFFSNAIVEGAKKFPNTEFVKVKGLHFSQEDAPDEMGNYIK
SFVERVLKNEQ
>DpsA
MASRNQSAIPLVTADEWWGKCKKVDVLGSKMSYYDSDPQNLSGKHTVVFLHGNPTSSYLW
RNVIPQVEPIARCLAPDLIGMGRSDKLASRSYRFLDHYRYLSAWFDALKLPEKITVVCHD
WGSALGFHWCNEHRGRLEAIVHMEGVYQPMTWEIFPDSMRDIFLALRSDAGEEMILKKNM
FIETILPLAIKRKLRQEEMDAYREPFKNPGEDRRPLLTFPRQIPIQGEGPEETVAIVTAY
HAWIKGTEDLPKFRILPTPGLFSEWGTGITKDWPNHKVVQVEGSHFFQEDSPIQTGDYIK
EFLSSIFK
>Sav4779
MPVQHILDSTMYHRESGTGVPIVFLHGNPTSSYLWRDVMPAVGSGRLLAPDLIGMGESGK
PALDYTFADHARYLDAWFDALDLRDVILVGHDWGGALAFDWAARHPHRVRGIAFTETIVK
PMAWAEFPEGGRELFRAIKTRGVGESMILDDNAFIEQGLPGSSATALTEGDLDVYRKPYP
TRESRLPLLRWPRSMPLGGEPADVVARIEAYDRWLKASVDVPKLLLTFAPGPGAMMHEGI
VAWCAANIAGLEIEHSEAVAGHHTPEDQPVLIARAISAWADRLGLRLS
>DatA
MTEKSPHSAFGDGAKAYDVPAFGLQIHTVEHGSGAPIVFLHGNPTSSYLWRHIFRRLHGH
GRLLAVDLIGYGQSSKPDIEYTLENQQRYVDAWFDALDLRNVTLVLQDYGAAFGLNWASR
NPDRVRAVAFFEPVLRNIDSVDLSPEFVTRRAKLRQPGEGEIFVQQENRFLTELFPWFFL
TPLAPEDLRQYQTPFPTPHSRKAILAGPRNLPVDGEPASTVAFLEQAVNWLNTSDTPKLL
LTFKPGFLLTDAILKWSQVTIRNLEIEAAGAGIHFVQEEQPETIARLLDAWLTRIAGN
>DmxA
MTTQKPADFPYPSHFADVLGSRMHYVEHGNGDPLLFLHGQPTWSYLWRKVLPELEGKGRL
IAVDLIGYGMSDKPDIPYDIDDHIRYLDGFIEALGLDRITIVCHDWGSFFGFHYAHRHPE
RIKGLAFMEAMLNPIPGYDAFDPQTRAFFQTLRSSQANAERMMMDENQFVENILPAMICR
PLERQELDAYRAPWTDRQSRRILCTFPQNLCIGKEPASVYRMQTAYIEWLGQTDLPKLLI
HAEPGFLIPAPAVDQYRQQLPNLETAFVGSGLHYIQEDQPQKIGQAIAQWMDRCGL
>DmlA
MSSKANPPQPVATAPKRSQIPILDSTMSYVEAGASGPTVLFLHGNPTSSHIWRNIIPHVA

PFGRCIAPDLIGYGQSGKPDIDYRFFDHVRYLDAFLDALDIRDVLLVAQDWGTALAFHLA
ARRPQRVLGLAFMEFIRPFERWEDFHQRPQAREMFKALRTPGVGEKLVLEDNVFVEKVLP
ASVLRAMSDDEMDVYRAPFPTPQSRKPVLRLPREMPIEGQPADVAAISAHDHRALRLSTY
PKLLFAGDPGALIGPQAAREFAAGLKNCSFINLGPGAHYLQEDHADAIGRAIASWLPEVV
LANQTDELA
>DbjA
MSKPIEIEIRRAPVLGSSMAYRETGAQDAPVVLFLHGNPTSSHIWRNILPLVSPVAHCIA
PDLIGFGQSGKPDIAYRFFDHVRYLDAFIEQRGVTSAYLVAQDWGTALAFHLAARRPDFV
RGLAFMEFIRPMPTWQDFHHTEVAEEQDHAEAARAVFRKFRTPGEGEAMILEANAFVERV
LPGGIVRKLGDEEMAPYRTPFPTPESRRPVLAFPRELPIAGEPADVYEALQSAHAALAAS
SYPKLLFTGEPGALVSPEFAERFAASLTRCALIRLGAGLHYLQEDHADAIGRSVAGWIAG
IEAVRPQLAA
>DbeA
MTISADISLHHRAVLGSTMAYRETGRSDAPHVLFLHGNPTSSYIWRNIMPLVAPVGHCIA
PDLIGYGQSGKPDISYRFFDQADYLDALIDELGIASAYLVAQDWGTALAFHLAARRPQLV
RGLAFMEFIRPMRDWSDFHQHDAARETFRKFRTPGVGEAMILDNNAFVERVLPGSILRTL
SEEEMAAYRAPFATRESRMPTLMLPRELPIAGEPADVTQALTAAHAALAASTYPKLLFVG
SPGALVSPAFAAEFAKTLKHCAVIQLGAGGHYLQEDHPEAIGRSVAGWIAGIEAASAQRH
AACRAKRART
>DhaA
MSEIGTGFPFDPHYVEVLGERMHYVDVGPRDGTPVLFLHGNPTSSYLWRNIIPHVAPSHR
CIAPDLIGMGKSDKPDLDYFFDDHVRYLDAFIEALGLEEVVLVIHDWGSALGFHWAKRNP
ERVKGIACMEFIRPIPTWDEWPEFARETFQAFRTADVGRELIIDQNAFIEGALPKCVVRP
LTEVEMDHYREPFLKPVDREPLWRFPNELPIAGEPANIVALVEAYMNWLHQSPVPKLLFW
GTPGVLIPPAEAARLAESLPNCKTVDIGPGLHYLQEDNPDLIGSEIARWLPAL
>Jann2620
MKRIRALATAATLAAGLAMPVAAQDTGGAQQPISAEFPFELQTVEVLGSNMAYVDTGDGP
VVLFIHGNPTSSYLWRNVIPHVAEDHRAIAIDLIGMGASDKPDIDYTFQDHYAHLEGFID
ALELTDITLVLHDWGGGLGTYYAANNSDNVRAIAMMEAAAPPALPIPDWAMVADQQTRET
FQAFRDPVMGPQIILEQNGFVEGLLPATILRTLSDAEMDAYRAPFPTPESRQPVLMWPNE
IPIEGTPARNVTVMEEVAAWLTTSEQPKLILYASPGLIWSPEVADFAARTFNNTEARFVG
AGIHFIQEDQPEAIGRNLSDWLRDRVTRGN
>DmmA
MASSSEFPFAKRTVEVEGATIAYVDEGSGQPVLFLHGNPTSSYLWRNIIPYVVAAGYRAV
APDLIGMGDSAKPDIEYRLQDHVAYMDGFIDALGLDDMVLVIHDWGSVIGMRHARLNPDR
VAAVAFMEALVPPALPMPSYEAMGPQLGPLFRDLRTADVGEKMVLDGNFFVETILPEMGV
VRSLSEAEMAAYRAPFPTRQSRLPTLQWPREVPIGGEPAFAEAEVLKNGEWLMASPIPKL
LFHAEPGALAPKPVVDYLSENVPNLEVRFVGAGTHFLQEDHPHLIGQGIADWLRRNKPHA
SSLE
>Aspnig-350639731
MTSYGYAATFVMSFTVCRLILFCSLLPLYLVKLSCNISPTLVRNFACLFSSQGRNRLVQE
AEEYRGQCESPAIAQCQGTYHFIEYNLYRVRRQGPGWMPDIPWADCTEEMAICGATVRFV
HQKPTLDGPADRRRPIIFLHGNPSWSYIWRNILPHLVDQGHEVYALDWLGHGRSDKATHP
ESVTFELHIHTLLQLFKVHNLRHAAIVAHDWGGCVALCAIPQLPQMACTRLLLLNSFLPP
RPDEWNLHYGLLYAIWFLSTGLLGGYVPESGVMRFMSPHLSQPEVDGYSAPYQSLPVATK
ASVFRFGHIVPVTPRFALHGLRQTRLWKLLEGLCGPRHFDNLSQQARLAKRGEEVRQSWR
AGLEKGNPDVAVVFGEQDPLLKLYKDILVDCIHPAHMVEWATQGIWLSGGGHYPMEEKAA
DISLLAEKLASQ

Příloha C

Obsah přiloženého CD

- `env/` – pracovní adresář pro běh frameworku s připravenými ukázkovými vstupy pro rodinu haloalkan dehalogenáz v podsložce `tmp/shared`.
- `lib/` – knihovny využívané ve zdrojovém kódu aplikace.
- `report/` – zdrojové soubory pro kompilaci této technické zprávy systémem LaTeX.
- `src/` – zdrojové soubory implementovaného nástroje v jazyce Java.
- `README` – návod na spuštění.