MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Vladimír Patík

Platforma pro hodnocení efektivity molekulární podobnosti

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Petr Škoda
Studijní program: Informatika
Studijní obor: Obecná informatika

Praha 2017

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Děkuji Mgr. Petru Škodovi za jeho pomoc a poznatky při zpracování bakalářské práce.

Tato práce vznikla za podpory projektů CERIT Scientific Cloud (LM2015085) a CESNET (LM2015042) financovaných z programu MŠMT Projekty velkých infrastruktur pro VaVaI.

Dále bych chtěl poděkovat rodině za podporu, kterou mě během psaní bakalářské práce poskytla.

Název práce: Platforma pro hodnocení efektivity molekulární podobnosti

Autor: Vladimír Patík

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Petr Škoda, Katedra softwarového inženýrství

Abstrakt: V bakalářské práci jsem navrhl a vypracoval platform pro testování ligand-based virtual screening (LBVS) metod, která umožňuje komunikaci pomocí webových stránek, spouštění testů na infrastruktuře Metacentrum nebo vzálených počítačích pro urychlení zpracování testů, přidávat vlastní datové sady a LBVS metody. K programu je přiloženo několik základních metod a z internetových zdrojů zmíněných v textu lze stáhnout dva balíky základních datových sad. Použiltenost platformy byla ověřena na dvou vzorových případech užití. Prvním byla replikace a ověření správnosti publikovaného experimentu. Druhým příkladem bylo porovnání metody založené na počtu atomů v molekule s metodami z knihovny RDKit.

Klíčová slova: chemická informatika molekulární podobnost ligand-based virtuální screening benchmarkování

Title: Platform for benchmarking efficiency of molecular similarities

Author: Vladimír Patík

Department: The Department of Software Engineering

Supervisor: Mgr. Petr Škoda, The Department of Software Engineering

Abstract: In the bachelor thesis I designed and developed a platform for testing of the ligand-based virtual screening (LBVS) methods, which allows communication via web sites, running tests on the Metacentrum infrastructure or remote computers for speeding up the processing of tests, adding own data-sets and LBVS methods. Several basic methods are attached to the program and two packages of basic datasets can be downloaded from the Internet sources mentioned in the text. The use of the platform has been verified in two exemplary use cases. The first goal was to replicate and verify the accuracy of the published experiment. The second example was the comparison of method based on the count of atoms in molecule with the RDKit library methods.

Keywords: chemoinformatics molecular similarity ligand-based virtual screening benchmarking

Obsah

Úvod

Molekuly jsou sloučeniny tvořené spojením více atomů za pomoci chemických vazeb. Významnou skupinou molekul jsou proteiny 1.1, které v našem těle zajišťují transport látek, ochranu a podporu buněk a zároveň umožňují řídit biologické a chemické reakce. O poslední zmíněnou úlohu se stará podskupina nazývaná enzymy 1.1.

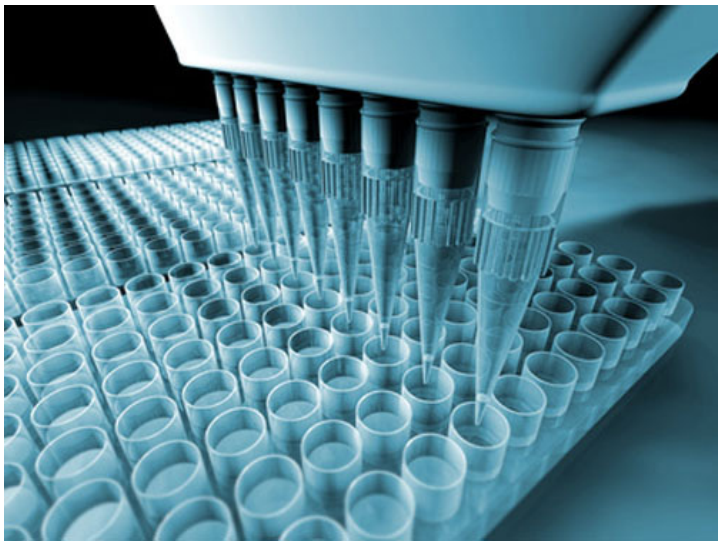
Příkladem enzymu v lidském těle je pepsin. Vyskytuje se v žaludku a štěpí proteiny obsažené ve stravě na jemnější cukry, aby mohly být později vstřebány skrz stěnu střeva do krevního oběhu.[1]

Vlivem okolního prostředí může docházet u enzymů k jejich dysfunkci nebo jejich nedostatku, což může způsobit hromadění toxických nebo nedostatek esenciálních látek v těle, a tím zapříčinit vznik nemocí. Farmaceutické firmy vydávají přes 50 mld. dolarů ročně na vývoj a výzkum, aby objevily příčiny těchto poruch. Díky tomu jsou schopny rozeznat charakteristické vlastnosti zkoumaných proteinů a vyrábět pro ně účinnější léky.[2]

Význačná skupina léčiv je založena na malých molekulách, které mají nízkou molekulární hmotnost, a tedy lehce pronikají do buňky. Uvnitř ní svou aktivitou ovlivňují jiné molekuly např. proteiny, jejichž změna struktury může zapříčinit změnu funkce a jejich chování dle požadavků výrobce léčiv.[3]

Základním způsobem hledání těchto látek je laboratorní zkoumání. Cílové proteiny trpící dysfunkcí se podrobují zkoušení reaktivnosti s malými molekulami, o nichž se domníváme, že by mohly vůči nim mít požadovaný efekt. Pokud tento efekt existuje vůči danému proteinu, považujeme je za aktivní proti danému proteinu.

Většinou se úspěšnost objevení aktivity pohybuje okolo 2% z celkového počtu testovaných látek. Aktivní malé molekuly se dále předávají k následnému testování pro zjištění detailů o jejich interakcích, možných specifických využitích či zlepšujících modifikacích. [5]



Obrázek 1: High throughput screening. [4]

Molekuly se testují v sadách, které mívají i milion molekul. Otestovat je při dnešní modernizaci může trvat dny až týdny. Při takovémto množství a několika procentní úspěšnosti skončíme u stovek tisíců látek s negativním výsledkem. Při započítání ceny 0.1 – 1$ za činidlo se dostaneme minimálně k 100 000$ za výdaje na jednu sadu.[6, 7]

Během testování se snažíme předejít vysokým finančním nákladům. Proto

existuje snaha omezit počáteční testovací množinu vstupující do laboratorního zkoumání, k čemuž nám napomáhá metoda zvaná virtual screening (VS) zakládající se na počítačovém předzpracování zkoumaných látek.

U metod patřících do VS se často stává, že na různých datových sadách fungují s jinou úspěšností. Jelikož do HTS vstupuje jen prvních $X\%$ z molekul otestovaných VS, i malé odchylky mohou způsobit velké snížení úspěšnosti celého procesu. Z tohoto důvodu je potřebná možnost jejich otestování na různorodých datech. Na internetu sice již existují testovací programy, ale složitost obsluhy, chybějící grafické rozhraní a komplikovaný přístup k předpřipraveným sadám práci s nimi stěžuje. Příkladem takového softwaru je open-source platforma od S. Rinikera a G. A. Landruma[8].

V bakalářské práci jsem navrhl a naprogramoval platformu, která poskytuje jednoduché a intuitivní ovládání pro testování VS metod. Pro zrychlení otestování metod nabízí paralelní spouštění na více počítačích či využití infrastruktury Metacentra. Všechny výsledky metod se automaticky analyzují základními vyhodnoceními AUC a EF, jejichž závěry lze později využít pro složitejší analýzy.

1. Chemoinformatická teorie

V následující kapitole jsou shrnuty základní poznatky o technikách zkoumání aktivity molekul vůči proteinům a stručné definice chemo-informatických pojmů, které se v tomto oboru používají.

1.1 Základních pojmy

Proteiny Proteiny jsou vytvořeny ze stovek až tisíců aminokyselin (molekul s karboxylovou a aminovou funkční skupinou) uspořádaných za sebou v dlouhých řetězcích. Aminokyseliny svým sekvenčním uspořádáním v řetězci určují výslednou primární strukturu proteinu a jeho funkci. V řetezci existují místa, na které se vážou molekuly zvané ligandy. Proteiny jsou základní stavební kameny buněk a jsou přítomné ve všech živých organismech (např.v lidském těle tvoří 17 až 20% hmoty).

Enzymy Enzymy jsou podskupinou proteinů. Vytváří v těle látky, které pomáhají urychlovat chemické procesy při nižší aktivační energii než je vyžadována, a tím ovlivňovat metabolismus a jiné biologické funkce. Enzymy mají složitou strukturu, do které jsou schopné napojit jiné sloučeniny, které nazýváme substráty, a tím urychlit chemickou reakci, které substráty mezi sebou podléhají. Tento děj nazýváme enzymatickou reakcí a probíhá v části zvané aktivní místo. Je to oblast strukturně velmi specifická a pouze určité látky, které mají doplňkovou stavbu, jsou schopny se sem navázat. Tato vlastnost dělá z enzymů jednoúčelové urychlovače. Jejich aktivita je ovlivňována množstvím přítomných agonistů a antagonistů v okolí a jeho fyzikálními a chemickými vlastnostmi.[9, 10]

Ligandy Ligand je typicky malá molekula, která se váže na vazebné místo cílového enzymu. Svým navázáním může změnit jeho funkci.

1.2 High throughput screening

High throughput screening (HTS) je jeden z principů objevování aktivních látek vůči cílovým proteinům. Při provádění HTS se vzorky látek umístěných v laboratorních zkumavkách podrobují testování na reakci vůči testovaným molekulám. Z tohoto důvodu HTS laboratoře musí mít zajištěnou fyzickou dostupnost zkoumaných látek, což prodražuje výsledné experimenty.

Obvykle dochází k hromadnému testování statisíců až milionů látek, aby se dosáhlo co nejlepší efektivity. Dříve trvalo postupným zkoušením interakce různých látek několik měsíců až let, ale díky modernizaci postupů např. robotickou automatizací v řízení a analýze jsou laboratoře schopny vyprodukovat velké množství informací o aktivitě zkoumaných molekul během několika dní až týdnů. HTS se nejvíce uplatňuje ve výzkumu farmaceutik, u validace cílových sloučenin a identifikace genů a proteinů, které mají biologický vliv. [11]

1.3 Virtual screening

Provedení HTS vyžaduje specifické požadavky na přístroje, lidské zdroje, skladování, nákup, logistiku a výrobu látek, které stojí laboratoře nemalé náklady. Z tohoto důvodu se přistupuje k doplňkové metodě nazývané virtual screening (VS), která testuje molekuly známých chemických struktur vůči cílovým proteinům in silico. Tato metoda se snaží předvídat ty molekuly, které s větší pravděpodobností budou více reaktivní. To pomáhá existovat menším laboratořím s malým množstvím molekulových sad a šetřit peníze velkým společnostem.

Existují dva přístupy k testování aktivity molekul. Prvním z nich je structure-based virtual screening (SBVS). U cílového proteinu musí být známa struktrura. Při její neznalosti lze využít porovnávání zkoumaných molekul k již zjištěným ligandům, které byly získány na základě laboratorního screeningu. Skupina zastřešující tyto metody se nazývá ligand-based virtual screening (LBVS), kterým se věnuji i v této práci.

I přes dekády výzkumů a velký pokrok v chemoinformatice existují problémy, které jsou stále řešeny a brání častějšímu používání LBVS metod v praxi. Příkladem mohou být:

1. Rozmanitost látek - množství vytvořitelných látek je velké a nalezení pouze biogenních molekul složité

2. Složitost strukury molekul - struktura před navázáním, po něm, různé její konformace

3. Výpočet podmínek reakce - spočítání přesné aktivační energie

4. Komplikovanost LBVS metod - nastavení parametrických metod

5. Různé úspěšnosti metod - VS metody mají na různých datových sadách různé úspěšnosti předpovídání aktivních molekul

Při špatných výpočtech dochází k mylnému předpovězení aktivity a snížení úspěšnosti nalezení aktivních molekul. I přes výše zmíněné nevýhody, které s sebou testování pomocí VS přináší, lze tuto metodu zařadit mezi užitečné nástroje při vývoji léků.

Jedno z možných využití je komplementární spuštění k HTS, kdy nalezneme aktivní molekuly, které se nám nepodařilo druhou technikou objevit. V některých výzkumech byla úspěšnost při objevování značně vyšší než u HTS (100 až 1000x vyšší). Bylo úspěšně předpovězeno několik nových ligandů 1.1 společně s jejich receptory, ke kterým se váží. [12]

Další možností je použití LBVS metod k rychlému odfiltrování molekul, které by pravděpodobně nebyly vůči testovanému proteinu aktivní, a tím dopomoci ke zmenšení vstupních datových sad do následného testování pomocí HTS.

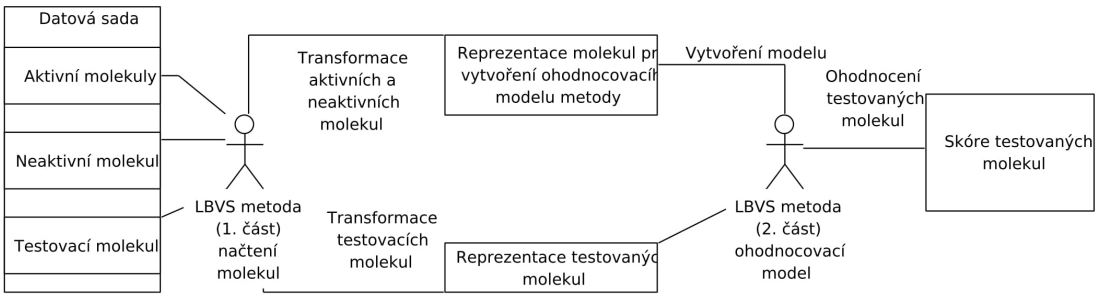
Protože se všechny látky vystupující ve VS syntetizují virtuálně, lze kromě využívání reprezentací fyzicky dostupných látek vytvářet i nové, ve skutečnosti nesyntetizované či prozatím nezakoupené sloučeniny. [12]

1.3.1 Ligand-based virtual screening

Hlavním úkolem ligand-based virtual screening (LBVS) metod je snaha o přiřazení hodnot (priorit) testovaným molekulám, které charakterizují jejich aktivitu vůči cílové látce. Toho se dosahuje vypočtením jednotlivých podobnostní (skóre) k poskytnutým referenčním aktivním ligandům a neaktivním molekulám. Základní myšlenka je vystavěna na Johnsonovo a Maggiorovo principu podobnosti vlastností, který tvrdí, že látky s podobnou chemickou strukturou mají i podobnou biologickou aktivitu. [13]

Ačkoliv principy při výpočtu skóre se u rozdílných LBVS metod mohou výrazně lišit, lze celkový průběh popsat diagramem 1.1.

Na vstupu metoda dostane datovou sadu, ve které jsou molekuly rozděleny do tří skupin: aktivní, neaktivní a testované molekuly. První dvě kategorie jsou tvořeny pozitivními a negativními příklady, které se využívají k vytvoření ohodnocovacího modelu metody, jehož úkolem je přiřazení číselných priorit molekulám z třetí skupiny. Činnost metody lze rozdělit na dva kroky (zobrazené na diagramu odděleně). V prvním kroku si metoda načítá datové sady a vytváří vlastní reprezentace molekul. V druhém kroku se vezmou postupně všechny molekuly z testovací sady a v rámci porovnávání molekul k prvním dvěma skupinám se určí podobnostním algoritmem výsledné podobnostní skóre. Hodnoty se seřadí a sestaví z nich prioritní žebříček.



Obrázek 1.1: Obecný popis výpočtu LBVS algoritmu

Tento žebříček pouze informuje o výpočítaných prioritách, ke kterým může uživatel metody přihlížet při předávání molekul do dalšího kroku testování (většinou HTS). Jelikož není obor hodnot u priorit omezen a zároveň není řečeno, od které příčky jsou molekuly s nějakou pravděpodobností aktivní, záleží na uživateli a jeho předchozích zkušenostech a znalostech, jaké množství molekul vybere do následného testování.

Často využívanou technikou při výpočtu ohodnocení molekuly je max-fussion, kdy metodě je poskytnuto více ligandů. Ke každému z nich vytvoří metoda samostatné skóre, ze kterých je vybráno to nejvyšší, které molekule při porovnávání bylo přiděleno.

1.3.2 Reprezentace a podobnosti LBVS metod

Reprezentace získána v načítávacím kroku určuje možnosti a postupy, při kterých se molekuly v porovnávacím kroku ohodnocují. Postupy můžeme rozdělit do několika skupin dle podobností založených na:

1. Superpozici molekul

   Hledání podobných látek je založeno na vzájemném porovnávání tvarů a pozic substruktur v molekulách. Reprezentací mohou být grafy či tvary sloučenin. Můžeme zkoumat pozice atomů molekul ve 2D i 3D prostoru. Podobnost se vypočítává např. největším společným podgrafem, superpozicí molekul či podobností vlastností tvarů látek. Příkladem takto pracujícího algoritmu je RASCAL. [14]

2. Statistikách molekul

   Metody založené na těchto podobnostech si nejdříve zjistí statistiky o testovaných molekulách. Na nich se v porovnávacím kroku provadějí srovnávací metriky. Příkladem výpočtu skóre je porovnávání překryvů histogramů získaných z molekul. [15, 16]

3. Deskriptorech molekul

   Deskriptory mohou reprezentovat jednorozměrnou (počty atomů od různých prvků v molekule) i vícerozměrnou (izometrie) strukturu sloučenin. Reprezentací jsou řetězce hodnot reprezentující přítomnost některé části látky nebo jejích vlastností. Patří sem početná kategorie řetězců pojmenovaných fingerprinty (viz sekce 2D fingerprinty1.3.3). Příkladem dalších fingerprintů mohou být atom pairs, topological torsion, BCUT deskriptory, pharmacore nebo elektrotopologické stavové indexy. Všechny jmenované se vytvářejí z informací získaných z 2D struktur. [17]

1.3.3 2D fingerprinty

Reprezentaci pomocí fingeprintů (FP) navrhla skupina Chemical information systems (CIS) pro zjednodušení vyhledávání cílových molekul v relačních databázích. Fingerprinty jsou zde reprezentovany n-bitovými řetězci. Na základě dotazu jsou odfiltrovány molekuly nesplňující přítomnost požadované hodnoty bitu na n-té pozici, čímž dojde ke zmenšení množiny pro úplné porovnání a konečnému zrychlení vyhledávácího algoritmu. V současné době jsou FP používány i pro podobnostní vyhledávání.[18]

Ve formátu fingerprint je molekula zastoupena n-členným řetězcem, kde proměnná n je buď pevně stanovena v metodě nebo volbou definována uživatelem. Členy mohou být čísla udávající počet výskytů nějakých sub- či superstruktur v reprezentované molekule nebo hodnoty 0 a 1 značící přítomnost či absenci těchto struktur. Sémantika bitů je dána funkcí, která se snaží vybírat příznaky, jenž by mohly mít největší potenciál ovlivnit aktivitu molekuly. Bitová pole fingerprintů jsou povětšinou řídká, jelikož zastupované látky obsahují jen část z použitých vzorů.[19] Fingerprinty lze rozdělit do kategorií [20] na:

- Slovníkové

  Mají již předpřipravenou sadu vzorů, které bity reprezentují. Za výhodu lze považovat, že vzory byly vybírány obecně se snahou o nejširší použitelnost. [21] Zároveň lze výsledky reprezentace jednoduše interpretovat na základě znalostí slovníku. Nevýhodou se může ukázat skutečnost, že použitý výběr

vzorů dostatečně nereprezentuje rozdílnost zkoumaných látek, čímž úspěšnost podobnostní funkce může vést k horším výsledkům.

Příklady: MACCS keys

- Hashované

  Nezakládají se na žadné předurčené množině vzorů, které by měly bity (indexy v poli) reprezentovat. Během vytváření fingerprintu se vyextrahují jím dané substruktury. Z každé z nich se hashovací funkcí vypočítá numerická hodnota, která určuje pozici bitu ve fingerprintu, který se nastaví na "1", nebo hodnotu indexu v FP, na kterém se uložená hodnota zvýší o jedna.

  Příklady: Daylight

1.3.4 Příklady fingerprintů a podobností

V této kapitole je uvedeno několik podobnostních metod a fingerprintů, které jsou obvykle používány (Tanimoto, MACCS keys), nebo jsou využity během experimentů.

MACCS keys fingerprints Molecular ACCess System structural keys nebo MACCS keys fingerprinty jsou definovány sadou 166 vzorů, jejichž přítomnost je v molekule kontrolována.

Atom pairs Atom pairs fingerprinty jsou tvořené z topologické vzdálenosti mezi dvěmi atomy v molekule.

Topological torsion fingerprints Topological torsion fingerprinty používají cesty délky 4.

Morgan fingerprints Morganovi fingerprinty jsou nekomereční implementací algoritmů ECFP a FCFP. V molekule se analyzuje okolí každého obsaženého atomu. Nastavuje se parametr počtu atomů, které jsou v okolí zkoumány. V pokusech je tento parametr u fingerprintu nastaven na 1 a 2.

BG fingerprints Fingerprinty délky 128, kde i-té číslo v pořadí označuje počet vyskytujících se atomů od í-tého prvku z periodické soustavy prvků v molekule. Publikováno v článku [22] pány A. Bendrem & R. C. Glenem.

Základní podobnosti

	Přítomen v X	Nepřítomen v X	Součet
Přítomen v Y	a	b	$a+b$
Nepřítomen v Y	c	d	$c+d$
Součet	$a+c$	$b+d$	$n=a+b+c+d$

Braun Blanquet similarity

$$BBS(X,Y) = \frac{a}{\max(a+b,a+c)}$$

Dice similarity

$$DCS(X,Y) = \frac{2 * a}{2 * a + b + c}$$

Tanimoto similarity

$$TS(X,Y) = \frac{a}{(a+b) + (a+c) - a}$$

Euclidean distance Pro X a Y různé fingerprinty stejné délky:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

2. Testování výkonnosti metod

LBVS metody můžeme charakterizovat jejich rychlostí, typem vstupních a výstupních dat nebo výkonností.

Při jejich testování nás především zajímá, jak dobře si stojí při seřazování aktivity testovaných molekul. Úroveň správnosti tohoto seřazení nazýváme výkonnost.

Během vývoje metod vznikla potřeba metody testovat mezi sebou a porovnávat vzájemnou úspěšnost na různých datových sadách. Během pokusů bylo zjištěno, že metody jsou na různých datových sadách různě výkonné. [8]. Důvodem mohou být nedostatečné a priori informace nebo nedokonalý ohodnocovací algoritmus.

Vybrat vhodnou metodu pro úspěšnou selekci vstupních dat vcházejících do HTS je důležitým rozhodnutím pro vědce a základním kamenem LBVS. Aby se zjistilo, která metoda je pro selekci zkoumané skupiny molekul nejvhodnější, spustí se různé metody na několika datových sadách podobných zkoumané. Pomocí analýz provedených na ohodnocených molekulách se následně můžou vědci rozhodnout, kterou z dostupných metod zvolí.

Aby uživatelům metod mohli jejich autoři poskytovat lepší varianty, potřebují vhodné prostředí, které jim umožní jejich metody otestovat a zanalyzovat. Aby věděli, zda vývoj postupuje k lepšímu a jak se metodě při ohodnocování daří ve srovnání s ostatními, mohou využít její otestování na různých datových sadách a vypočtění analýz na jejích výsledcích. Díky různorodosti datových sad, může autor získat přehled, které vlastnosti výpočetního algoritmu lze zdokonalit.

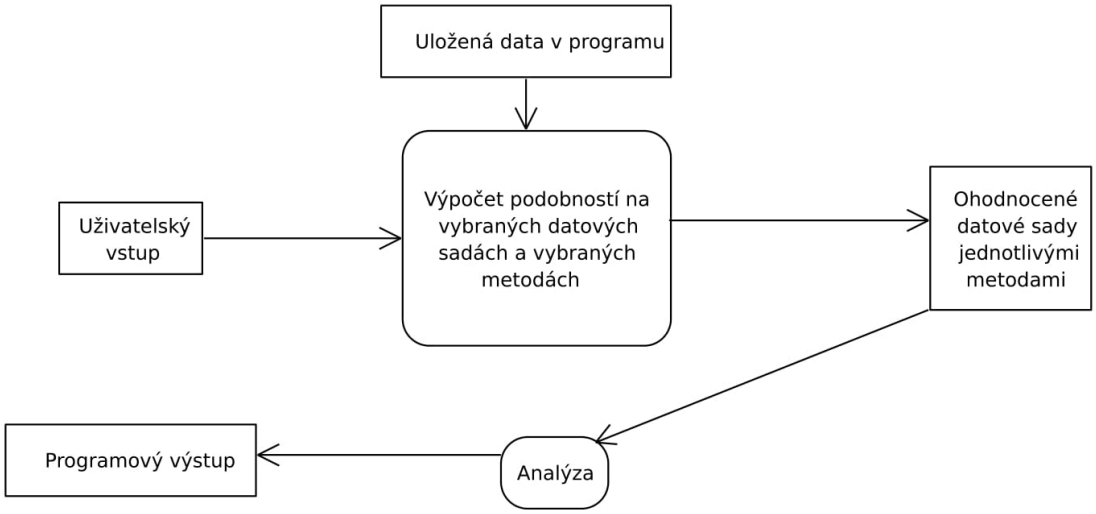
Tabulka 2 uvadí vstupní, výstupní a očekávaná předpřipravená data při testování metod a datových sad z pohledů autorů a uživatelů metod.

	Uživatel metody	Autor metody
Vstupy	Testovací sada Nové metody	Testovaná metoda Nové datové sady
Uložené data	Metody	Metody Datové sady
Výstupy	Porovnání metod na dodané testovací sadě	Srovnání dodané metody s ostaními metodami na různých datových sadách

Diagram 2.1 zobrazuje obecný princip testování LBVS metod. Testovací program obsahuje databázi, ve které má uloženy data dle potřeb uživatelů či autorů metod. Ze vstupu si přečte data, která dle jejich typu zpracuje, zanalyzuje a vydá v přehledném výstupu pro uživatele programu.

2.1 Testovací datové sady

Datová sada je soubor molekul, která se používá při testování jejich aktivity. Datové sady můžeme získat z laboratorních pokusů. Informace v nich obsažené poskytují přehled o zjištěných aktivitách testovaných látek vůči proteinu, proti kterému byly testy na reakci při pokusu prováděny.



Obrázek 2.1: Postup při testování výkonnosti LBVS metod

Na Internetu existují webové databáze (Zinc, PubChem), které shromažďují poznatky z pokusů nebo elektronické popisy molekul a poskytují je širší veřejnosti. Některé z nich jsou otevřeny zájemcům zcela zdarma.

Principy aplikované metodami jsou rozdílné. Aby se došlo ke zjištění jejich výkonnosti, jsou metodám při testování předkládány různorodé typy datových sad.

Datovou sadu (DS) dále rozděluje *výběr* na trénovací (TR), validační (VA) a testovací sadu (TE). Tyto sady jsou vzájemně disjunktní.

Trénovací sada slouží k vytvoření modelu, validační většinou k předejití jeho přetrénování či úpravě parametrů při učení. Testovací sada tvoří množinu dat, které jsou ohodnocovány vytvořeným modelem.

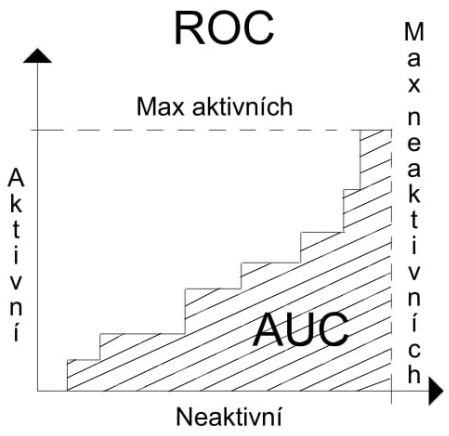
2.2 Analýza výstupních dat

Výstupní data metody jsou tvořena ohodnocením molekul z testovací sady. Jejich seřazení dle přidělených hodnot odpovídá jejich uspořádání podle předpokládáné aktivity vůči proteinu. Abychom získali alespoň částečný pohled na výkonnost metod, testujeme jejich výsledky analytickými nástroji.

V následujícím odstavci je uvedeno jen několik nejpoužívanějších přístupů k analýze LBVS metod.

Základním nástrojem pro vyhodnocení výkonnosti LBVS metod je ROC (Receiver operating characteristic) křivka 2.2, kterou zobrazujeme v 2D grafu s osou X značící počty neaktivních molekul a osou Y udávající počet aktivních. Při vytváření křivky čteme molekuly ze vstupu a posouváme se dle jejich aktivit po osách. Jelikož porovnávat grafy je složité, využívá se ke srovnávání metod plocha pod ROC křivkou jmenovitě AUC (area under receiver), jejíž hodnoty jsou mezi 0.0 (všechny neaktivní molekuly byly upřednostněny před aktivními) a 1.0 (opačný tedy optimální případ). Nevýhodou ale i výhodou oproti dále zmíněným metrikám je, že výsledek ROC (AUC) závisí na celé množině testovaných molekul z datové sady. Tento problém je znám pod názvem "early recognition"(ER).

Problém odstraňuje technika EF (the enrichment factor), která vyhodnocuje



Obrázek 2.2: ROC křivka/ AUC plocha

koncentraci aktivních molekul v prvních X sloučeninách v pořadí určeném metodou. Proměnná X je vypočítána jako $N * \alpha$, kde N je počet molekul v testované sadě a α je stanovena uživatelem v definičním oboru mezi 0 a 1.

Nevýhodou EF je silná závislost na poměru aktivních a neaktivních sloučenin v DS.

Další postup, který je robustní vůči ER problému, se nazývá BEDROC (the Boltzmann-enhanced discrimination of ROC). Lze jej interpretovat jako pravděpodobnost, že nějaká aktivní molekula bude upřednostněna před náhodně vybranou sloučeninou pomocí exponenciálního rozdělení s parametrem α. Kvůli jedinému parametru závisí citlivost techniky BEDROC na jeho volbě, který by měl být optimálně volen $n/N * \alpha << 1$. Z názvu je patrné, že jde o vylepšenou verzi ROC.

Poslední technikou je RIE (robust initial enhancement), která je sice taktéž odolná vůču ER problému, ale na druhou stranu její maximální hodnota je závislá na poměru aktivních a neaktivních molekul.[23, 24]

3. Uživatelská dokumentace

Virtual screening benchmark (VSB) je platforma sloužící ke spouštění testů, ukládání jejich výsledků, LBVS metod a datových sad. Mimo toho poskytuje základní nástroje k provedení analýzy proběhlých testů a jejich grafickému zobrazení. Platforma pro možnou interoperabilitu poskytuje export dat do formátu CSV.

Rozhraní programu je zprostředkováno skrz lokální webové stránky, příkazové soubory a konzoli.

Pro zrychlení výpočtů časově náročných algoritmů nabízí platforma spouštění testů na infrastruktuře Metacentrum či vzdálených počítačích definovaných uživatelem.

Pro možnost vytvoření více instancí programu na různých počítačích umožňuje program přesouvat mezi nimi části jejich lokálních DB.

K analýze výsledků metod se využívá knihovna RDKit, čímž je k dispozici i uživateli platformy.

Vysvětlení postupů pro práci s platformou se soustředí na využití webového rozhraní. Ostatní možnosti jsou uvedeny v záložce *dokumentation* na webových stránkách.

3.1 Instalace

Instalace základních závislostí

Kroky označené znakem * jsou volitelné. Instalace přídavných knihoven je prováděna pomocí nástroje PIP. Všechny příkazy jsou zadávány na operačním systému Windows 10 v příkazovém řádku aplikace PowerShell.

1. Python

   Pro spuštění VSB je vyžadováno prostředí jazyka Python[1] minimální verze 3.5 (instalace[2].

2. PIP

   Python verze 3.5 má již nástroj PIP předinstalovaný. Aktualizaci provedete příkazem

   ```
   > python -m pip install --upgrade pip
   ```

3. RDKit

   Postup instalace chemoinformatické knihovny je k nalezení na oficiálních stránkách[3].

4. FLASK

[1] https://www.python.org/
[2] https://wiki.python.org/moin/BeginnersGuide
[3] http://www.rdkit.org/docs/Install.html

Microframework Flask[4] pro vytvoření webového serveru a zprovoznění grafického rozhraní programu.

```
> pip install Flask
```

5.* PyMySQL

Ovladač PyMySQL[5] zajišťující komunikační agendu s MySQL serverem. Pouze při využití MySQL serveru.

```
> pip install PyMySQL
```

6.* MySQL server

Platforma poskytuje kromě ukládání informací do SQLite embedded databáze i ukládání do MySQL databázového serveru (v. 4.1 a vyšší). Postup instalace a instalační soubory lze nalézt na jejich webových stránkách [6]. Program si databázové schéma inicializuje sám. Přihlašovací údaje k přístupu se vyplňují při vytváření hlavního konfiguračního souboru.

Závislosti pro Metacentrum, vzdálené počítače

Jednou z nabízených možností, kde lze spustit testy, je infrastruktura Metacentrum[7]. Metacentrum je virtuální organizace poskytující výpočetní, licenční a úložné zdroje pro akademickou obec. Podmínkou využívání zdrojů je tamější registrace. O ni si může zdarma zažádat každý pracovník, zaměstnanec nebo student, který přísluší k akademické instituci v ČR a splní pravidla využití. Když je jeho přihláška odsouhlasena, jsou mu zpřístupněny všechny nabízené funkce.

Druhou možností je využití vlastního gridu vzdálených počítačů, na kterých mohou být spuštěny úlohy zadané platformě.

Uživatel je povinnen zajistit všechny dále zmíněné závislosti pro správné vykonávání programu.

- Vzdálená infrastruktura
  - Python verze 2.7 a vyšší
  - Operační systém Windows nebo Linux
  - Jiné závislosti dané zkoušenou metodou

  První dva požadavky na vzdálenou infrastrukturu jsou již Metacentrem splněny.

- Lokální počítač
  - Základní instalace VSB

[4] http://flask.pocoo.org/docs/0.12/
[5] https://pymysql.readthedocs.io/en/latest/index.html
[6] https://www.mysql.com/
[7] https://metavo.metacentrum.cz/cs/

– Ovladač PySFTP

Ovladač PySFTP zajišťuje mezipočítačovou komunikační agendu. Nainstulejete ho příkazem

```
> pip install pysftp
```

Instalace aplikace

Přesuňte se do složky, kde chcete aplikaci VSB uložit. Pokud máte nainstalovaný nástroj GIT, lze program stáhnout zadáním příkazu v GIT Bash příkazovém okně

```
> git clone
https://bitbucket.org/atipak/vsb.git
```

Jinak stáhněte zdrojové soubory z repozitáře[8] na bitbucket.org v ZIP archivu a rozbalte do aktuálního adresáře.

Vstupní datové sady

Ke spuštění a správnému běhu programu je nutné stáhnout jeden z nabízených balíků datových sad, které obsahují základní vstupní data pro testy metod.

Prvním balíčkem *ZDS* velikým 62 MB je množina vybraných datových sad z portálu PubChem sloužící k demonstračním ukázkám programu. K automatickému importu slouží skript *support_scripts/download_start_db.py*. Jeho spuštěním dojde k vytvoření základních složek a datových souborů v souborové databázi a zároveň ke stažení datových sad z Internetu.

Druhým balíčkem jsou state-of-art datové sady *SOAS* zabírajícím po stažení 14 GB (po vložení 55 GB). [25] Skript k jejich stáhnutí i s návodem lze nalézt v repozitáři[9] na Githubu. Po stažení datových sad do zvolené dočasné složky je lze importovat pomocí skriptu *support_scripts/import_external_db.py*. Po spuštění si skript v menu vyžádá cestu ke složce *data/datasets*, ve které se stažené datové sady nachází. Vkládání sad probíhá automaticky. Z důvodu velkého množství dat může trvat déle jak 1,5 hodiny. Po jeho skončení můžete dočasnou složku vymazat.

Konfigurační soubory

Hlavní konfigurační soubor Ke správnému fungování programu je nutné mít validní konfigurační soubor. Pro jeho vytvoření využijte skript *create_config_file.py* ve složce *support_scripts*. Pokud máte nainstalovaný MySQL databázový server nebo účet na infrastruktuře Metacentrum, budete během vytváření souboru dotázáni na přístupové údaje. Pokud se umístění Vaší souborové databáze nenachází ve složce *database*, musíte během dialogu zadat novou cestu k adresáři se vstupními datovými sadami. Konfigurační soubor můžete vytvořit buď ve složce *configurations*, odkud se implicitně data čtou, nebo ve Vámi zadané

[8] https://bitbucket.org/atipak/vsb/downloads/

[9] https://github.com/skodapetr?tab=repositories

složce pod Vámi zadaným jménem, pokud budete spouštět aplikaci s explicitně zadaným konfiguračním souborem např. z důvodu více konfigurací.

Konfigurační soubory pro spouštění testů na vzdálených počítačích Program dovoluje vytvořit více konfiguračních souborů, které v sobě ukládají informace o gridech, na kterých lze spustit testy v rámci možnosti spouštění testů na vzdálených počítačích. Tyto soubory se vytváří přímo v aplikaci v konzolovém rozhraní pod možností *add -> configurations*.

3.2 Spuštění

VSB se spouští pomocí skriptu *src/main.py*. Existují tři režimy, ve kterých lze aplikaci spustit.

Přepínač -cPath určuje cestu k explicitně zadanému konfiguračnímu souboru aplikace (zadává se pouze v případě, pokud se konfigurační soubor nenachází ve složce *configurations* s názvem *main_conf.json*).

- Automatický

```
> python main.py -a -file vstup.json
  [-cPath cestaKeKonfiguraci.json]
```

-a ... argument označující automatický režim
-file ... cesta k souboru s příkazy pro program

- Interaktivní

```
> python main.py -i [-cPath cestaKeKonfiguraci.json]
```

```
> python main.py
```

-i ... argument označující interaktivní režim

- Grafický

```
> python main.py -g [-cPath cestaKeKonfiguraci.json]
```

-g ... argument označující grafický režim

Pokud je soubor spuštěn bez zadání jakýchkoliv přepínačů, automaticky se přepne do interaktivního módu.

Při spuštění aplikace v grafickém režimu se webové stránky nachází na adrese *http://127.0.0.1:5000/*, pokud během vytváření konfiguračního souboru nebyl cílový port webových stránek aplikace měněn.

3.3 Pracovní skupiny, datové sady, metody

V následující sekci jsou popsány postupy pro vkládání dat pomocí webových stránek platformy a jejich základní popisy. Aby uživatel programu mohl plně manipulovat s webovými stránkami, musí nejdříve přihlásit nebo registrovat pracovní skupinu. Formulář pro její vytvoření se nachází na domovské stránce.

Pracovní skupina

Pracovní skupina spouštějící test slouží k selekci dat při vyhledávání v SQL databázi. *Group name* musí být neprázdný libovolný textový řetězec.

Nepovinné doplňující informace k pracovní skupině mohou obsahovat libovolný text, který si přejete u této položky v databázi uchovat.

Vždy existuje skupina "host", která je uložena pod identifikačním číslem 1. Můžete ji využít, pokud se nechystáte test nijak označit.

Metody

Metody lze přidat v hlavním menu *Method -> Add*.

Několik funkčních LBVS metod lze nalézt ve složce *test_func* v kořenovém adresáři aplikace VSB. Při prvním spuštění programu jsou do něj tyto metody automaticky vloženy jako počáteční data.

Program přijímá dva typy metod. Metody obecného typu (General), které lze spustit jakýmkoliv příkazem, a metody typu Python, jenž se skládají ze dvou nezávislých funkcí *loading* a *scoring* a jsou programované v jazyku Python. Podrobnější informace se nalézají v programátorské dokumentaci 4.1.

Datové sady

Datové sady lze přidat v hlavním menu *Data-set -> Add*.

Uživateli je poskytnuta možnost přidat si do programu své vlastní datové sady. Vzorový soubor, který datovou sadu definuje, se nachází ve složce *examples* pod jménem *import_dataset_example.json*. S datovou sadou lze přidat i výběry. Příklad souboru s výběrem se nalézá v téže složce pod jménem *import_selection_example.json*.

Během vkládání se kontroluje, zda všechny molekuly definované v datové sadě jsou obsaženy i v přiloženém SDF souboru. V případě, že nějaká molekula v SDF informacích chybí, datová sada se nepřidá. Při vkládání výběru se kontroluje, zda výběr obsahuje pouze molekuly zapsané v definici datové sady.

3.4 Spouštění testů

Program nabízí tři způsoby spuštění testů. Všechny volby lze nalézt v menu webových stránek *test -> Launch*. K výpisu podrobnějších informací o běhu platformy a testů se používá konzole, ve které byl program spuštěn.

Lokální počítač

Při volbě této možnosti dojde k zahájení testů na počítači, kde je aplikace VSB spuštěna. Lze vybírat ze sekvenčního a paralelního režimu. Maximální počet spuštěných testování na výběrech v jednom okamžiku je roven počtu dostupných jader v procesoru, nebylo-li během vytváření hlavního konfiguračního souboru stanoveno jinak. Jelikož zde není žádný overhead zaviněný přenosem dat a komunikační agendou, hodí se tento typ zkoušení pro rychlé algoritmy.

Varování: Paralelní volba způsobí nahodilé prokládání výpisů informací od různých testů do konzole.

Vzdálené počítače

Pokud máte k dispozici větší množství počítačů, které splňují podmínky stanované platformou pro vzdálené počítače, můžete je využít ke spuštění testu. Na webových stránkách Vám budou nabídnuty uložené konfigurační soubory s popisem vzdálených počítačů. Tyto soubory lze vytvořit v konzolovém rozhraní pod možností *add -> configurations*.

Na každém vzdáleném stroji je metoda spuštěna na různém výběru. Pokud množství výběrů přesahuje počet dostupných počítačů, jsou stroje využívány opakovaně. Testy jsou ukládány na cílových počítačích do složky *VSB_folder* umístěné v domovském adresáři. Pokud vše proběhne v pořádku nebo nastane jedna ze známých výjimek, jsou soubory z cílových počítačů smazány, jinak jsou ponechány na místě.

Metacentrum

Spouštění testů na infrastruktuře Metacentrum je rozšířením způsobu spouštění na vzdálených počítačích. Díky sdíleným uložným prostorům může být přenos dat zrychlen přenesením všech dat najednou. Pro každý výběr se registruje právě jedna úloha. Standardní a chybové výstupní soubory jsou na Metacentru ukládány do adresáře příslušného testu nacházejícího se ve složce *database/results*. Všechny informace o proběhlých testech a vstupní data jsou ponechány pro archivaci na infrastruktuře. Pokud uživatel nemá zájem si tyto údaje ukládat, musí je ručně vymazat. Kopie souborů výstupů metod jsou přeneseny na lokální počítač. Testovací data jsou na Metacentru ukládány do složky *VSB_folder*.

Při spouštění testů na Metacentru budete na webových stránkách aplikace vyzváni k zadání předního uzle (počítače), se kterým bude prováděna komunikace. Při velkých přenášených objemech dat může výběr bližšího umístění dopomoci ke zrychlení jejich přenosu. Další požadovanou informací jsou přihlašovací údaje do služby Metacentrum, které se pro potřeby testu ukládají do dočasné složky *web_interface/temp*. V případě ukončení aplikace standardním způsobem se data vymažou.

Upozornění

U spouštění na vzdálených systémech musíte počítat s možným zpožděním způsobeným přenosem dat po síti, alokováním prostředků pro vykonávání testu a mezipřístrojovou komunikací. Je třeba mít na paměti, že tyto události trvají

delší dobu a při krátkém trvání běhu testovaných metod na jednom výběru mohou zabírat podstatnou část vykonávání testu.

3.5 Výsledky metod a analýza dat

Získatelná data

Aplikace si ukládá většinu pracovních souborů a výstupních dat. Tyto data jsou zobrazitelná v tabulách a grafech na webových stránkách aplikace.

Metody lze stáhnout z webových stránek rozkliknutím jejich detailu v přehledu zobrazitelným po kliknutí na položku *Method -> List*.

Datové sady mají přehled umístěn v *Data-set -> List*.

Výběry od datové sady mohou být staženy rozkliknutím jejich detailu v přehledu, který je umístěn u detailu každé datové sady.

Informace o proběhlých testech se získávají pomocí menu *Test -> Results*.

Analyzování dat

Analýza dat se provadí v položce *Analysis* v hlavním menu na webových stránkách. Položky označené * jsou zpracovávány samostatným skriptem *external_analysis.py*, kterému se předává vstupní soubor vygenerovaný webovými stránkami.

Následující seznam představuje všechny možnosti analýzy, které lze na webových stránkách udělat.

- Compare methods/selections

  Compare methods on selection porovná uživatelem vybrané metody na všech výběrech, které se zúčastnily jejich testování. *Compare selection on methods* zobrazí seznam základních analýz výkonnosti metod, které byly na zvoleném výběru spuštěny.

- Compare methods *

  Na zvolených metodách provede porovnání výběrů (při zaškrtnutí checkboxu *Collapse to data-sets* datových sad). Zobrazí přehled metod s jejich dosaženým skóre a úspěšností na výběrech. Tabulka *balances* určuje rozdíl hodnot získaných metodou a hodnot nejúspěšnější metody. U výběrů (datových sad) zobrazí přehled analýz a molekul.

- Compare selections

  Výsledek rozboru je stejný jak u *Compare methods*, avšak místo metod volí uživatel výběry.

- Compare data-sets and methods

  Porovnání základních vyhodnocení na uživatelem zadaných metodách a datových sadách. Počet molekul, na kterých jsou základní analýzy provedeny, lze změnit ve formuláři po zobrazení úvodních výsledků, které jsou vždy dělány na všech molekulách. Z důvodu náročnějších výpočtů jsou základní analýzy na prvních K molekulách prováděny skriptem *external_analysis.py*.

- Export test analysis to CSV

  Exportuje výsledky základních vyhodnocení testů do formátu CSV.

- Show external analysis

  Přijímá soubor vygenerovaný skriptem *external_analysis.py* a zobrazí přehledně výsledky v něm uložené na webových stránkách.

- Clusters on collapse fingerprints *

  Vytvoří klastr na metodách. Jsou využity fingerprinty tvořené řetezcem dvojic (aktivní, neaktivní). Číslo *Windows count* říká počet dvojic a *Window size* určuje velikost posloupnosti, ze které se dvojice tvoří.

  Při menší velikosti okna a větším počtu dvojic sloučí dřív metody, jejichž žebříčky ohodnocených molekul jsou si podobnější. Řeší drobné změny ve skóre.

- Clusters on subtest analysis *

  Při vytváření klastru na metodách jsou využity fingerprinty tvořené základními vyhodnoceními testů.

  Sloučí dříve metody, které mají podobnější výkonnosti.

- Same molecules for first K *

  Vypočítá počet stejných molekul pro prvních K na výsledcích testů provedených metodami zvolených uživatelem.

- Known actives for data-set *

  Pro zadanou datovou sadu vytvoří přehled prováděcích molekul v prvních K. Ke každé known active zobrazí počet provedených a jejich jména.

- Cluster on fingerprints created by known actives on same places

  Vytvoří klastr na metodách. K výpočtům jsou využity fingerprinty tvořené stejnými known actives na stejných místech.

  Pro vzdálenější metody se stejnou výkonností odhalí, že k objevení ligandů používají jiné known actives. Při souběžném použití takovýchto metod se zvyšuje šance na objevení různých aktivních molekul.

- Cluster on fingerprints created by count of molecules which are leaded of known actives in first K molecules

  Vytvoří klastr na metodách. K výpočtům jsou využity fingerprinty tvořené počty molekul, které byly provedeny stejnými known actives v prvních K molekulách.

  Stejné použití jako v předešlé metodě, ale s jiným postupem při výpočtu klastru.

- Cluster on fingerprints created by relative order of active molecules

  Vytvoří klastr na metodách. K výpočtům jsou využity fingerprinty tvořené relativní pozicí aktivních molekul.

Sloučí dříve metody, které ohodnocují podobně, ale jedna z nich má ligandy posunuté o X pozic dolů v žebříčku. Nebo jsou-li malé změny pozic aktivních molekul v žebříčku.

- Cluster on fingerprints created by count of inactive molecules before i-th active one

  Vytvoří klastr na metodách. K výpočtům jsou využity fingerprinty tvořené počty neaktivních molekul před í-tou aktivní.

  Podobné použití jako v předešlé metodě.

- Cluster on fingerprints created by differences between scores of two following active molecules

  Vytvoří klastr na metodách. K výpočtům jsou využity fingerprinty tvořené relativními rozdíly v přiděleném skóre vždy mezi dvěmi následujícími aktivními molekulami.

  Sloučí dříve metody, jenž ohodnocují molekuly hodnotami, které jsou v podobné relativní vzdálenosti.

U všech možností klastrování je nabízeno jejich vykonání na celých datových sadách místo výběrů. Tato možnost se aktivuje zaškrtnutím checkboxu *collapse to data-sets* při vytváření zadání.

3.6 Rozhraní programu

Všechny tři poskytované rozhraní dokážou obsluhovat základní funkce programu. Konzole je důležitá při instalaci platformy či přímé manipulaci s databázemi, ale není velmi uživatelsky přívětivá. Z tohoto důvodu jsem v demostračních příkladech použití programu zvolil webové stránky.

Konzole

Konzole je uživatelsky nejsilnější nástroj pro ovládání programu. Poskytuje víc servisních funkcí než grafické rozhraní (příkladem je přesun databází či vytváření konfiguračních souborů). Konzole nekotroluje 100% validitu příkazů a jejich špatným zadáváním lze způsobit problémy v databázi (Spuštění 2 testů na stejných metodách s přímým zadáním cesty k souboru zaviní duplicitní záznam metod v SQL databázi pod jiným identifikátory).

Souborové příkazy

Pokud chcete vyplnit více požadavků najednou, které mají stejné parametry nebo máte-li k nim předpřipravené dotazy, pak máte možnost využít řízení programu pomocí souboru. Příkazy se zapisují do souboru dle specifikací uvedených v předloze *input.json* ve složce *examples* v kořenovém adresáři aplikace VSB. Ke každému typu příkazu je v souboru uveden komentář, který definuje povinné položky, jejich možné obsahy a výsledek příkazu. Soubor má strukturu dle příkladu 3.1.

```
{
   "test": {
      "comment": [ ... ],
      "0": {
         "d/s": [ ... ],
         "optimalization": "yes",
                  ...
         "parallel_selections": "False"
      },
      "5": {
         "output_file": "..\\res.json",
                  ...
         "user_id": 1
      }
   }
   "comment": [ ... ],
   "metadata": {
      "ignore":[ "comment", "1"],
      "parallel_tests": "False"
   }
}
```

Obrázek 3.1: Struktura příkazového souboru

Upozornění ID u user_id rovno jedné odpovídá identifikátoru pro vždy dostupného obecného uživatele *host*.

Položky s klíčem "comment"lze vynechat. V příkladovém vstupu slouží pouze pro okomentování struktury a obsahu.

Do testování budou zařazeny všechny testy, které jsou zapsány v položce "test"a jejich identifikátory nejsou uvedeny v "ignore".

Identifikátor testu (např. "5") může být jakýkoliv neprázdný řetězec uzavřený v uvozovkách. Jeho název se již dále nepoužívá a při vykonávání testu bude ztracen.

Při špatné manipulaci dochází ke stejným problémů, jenž byly zmíněny u konzolového přístupu.

3.7 Případy užití programu

K demonstraci webového rozhraní využiji dva případy užití. Prvním z nich je spuštění testů na Metacentru, druhý z nich popisuje spuštění testu na vzdálených počítačích. Aplikace je již nainstalovaná dle sekce 3.1(podsekce *Instalace základních závislostí a aplikace*), obsahuje balík *SOAS* a vytvořený hlavní konfigurační soubor, jenž jsou popsané v téže sekci.

3.7.1 Spuštění testů na Metacentru

Popis situace Uživatel chce zmenšit testovací množinu, která by měla vstupovat do HTS experimentu. Potřebuje proto vybrat metodu, která by byla vhodná pro správné seřazení jeho testovaných molekul. Výsledky chce získat co nejdříve. Jsou nainstalované rozšířené závislosti požadované Metacentrem (3.1 podsekce Metancetrum, vzdálené počítače). Uživatel patří k vědecké organizaci, a proto si mohl zařídit účet na infrastruktuře Metacentrum. V platformě se již nacházejí LBVS metody distribuované společně s programem.

1. Přihlášení

   Na hlavní stránce programu jsou dva formuláře. Jeden pro registraci nového uživatele, druhý pro přihlášení. Uživatel nepotřebuje testy oddělit, přihlásí se proto pod univerzálním profilem *host* kliknutím na tlačítko *Log in like a host*.

2. Vložení vlastních datových sad

   Uživatel má k dispozici datové sady, které jsou podobné testované z experimentu a na kterých je potřeba metody ověřit. Proto předělá jejich definice do formátu (*examples/import_dataset_example.json*). Ke každé DS

připraví SDF soubor obsahující z ní všechny molekuly. Takto připravené DS se postupně vloží pomocí položky *add* v menu *data-set*. Jméno datové sady je určeno jménem definujícího souboru, který vkládáme.

Definice datových sad lze stáhnout z databáze Pubchem[10]. Po kliknutí na vybranou datovou sadu na navrhované stránce, lze v sekci *Data table* stáhnout její definici souboru typu CSV (Data table (All)). Pomocí skriptu *fromCSVtoJSON* ve složce *support_scripts* převést do formátu přijímaného platformou. Skript *extractCIDS* vytvoří seznam indentifikátorů do souboru *cids.txt*. Pro získání SDF definic molekul z datové sady se tento soubor zadá jako vstup na stránce PubChem Download Service[11].

3. Vložení výběrů

   Z každé nově přidané datové sady uživatel vytvoří výběry, které mají strukturu (*examples/import_selection_example.json*). Tyto výběry postupně přidá nalezením příslušné datové sady v položce *data-set -> list*, rozkliknutím jejího detailu a kliknutím na tlačítko *Add new selection* v záložce *Selections*.

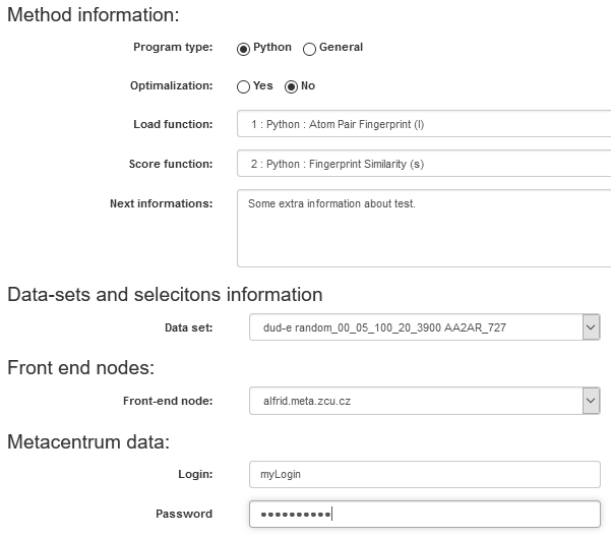
4. Spuštění testů

   Spouštění testů se děje pomocí menu *Test -> Launch*. Jelikož uživatel chce spustit testy na Metacentru, zvolí položku *Metacentrum* v submenu. Pro každou metodu a datovou sadu spustí jeden test.

   V oddíle *Method informations* vyplní informace o metodě, kterou chce testovat. Příklad na obrázku pod textem.

   V části *Data-sets and selecitons informations* uživatel zvolí datovou sadu, na které chce test spustit. *Front-end node* určuje přední uzel, se kterým bude probíhat komunikace. Zadání testu probíhá v Plzni, proto zadá výpočet na tamějším gridu, a tudíž vybere *alfrid.meta.zcu.cz*.

   V poslední části zadá své přihlašovací údaje k Metacentru a odešle formulář tlačítkem *Submit*.



[10]https://www.ncbi.nlm.nih.gov/pcassay?db=pcassay&cmd=search&term=(%231)%20AND%20(pcassay_pccompound_active[filt])&loc=s_frm

[11]https://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch.cgi

5. Kontrola testů

   Seznam nedoběhnutých a ještě nezkontrolovaných testů je udržován v přehledu na stránce *Test -> Running*.

6. Zobrazení výsledků po doběhnutí všech testů

   Cílem je zanalyzovat výsledky testů. Jedna z poskytovaných možností je zobrazení kartézkého součinu mezi vybranými metodami a datovými sadami. Pro tuto volbu vybereme v menu *Analysis* položku *Compare datasets and methods*. Nejdříve se zobrazí webová stránka s výběrem datových sad, kde uživatel zaškrtne pomocí check-boxů své vložené sady. Po kliknutí na tlačítko *Next* se dostane k volbě metod. Když výběr ukončí a potvrdí, začne se zpracovávat analýza, jejíž doba trvaní se odvíjí v závislosti na množství vybraných položek. Po načtení stránky se zobrazí graf znázorňující výsledky analýzy. Lze zobrazit průměrné výsledky AUC a EF. Metoda s nejlepšími průběhy je pravděpodobně nejvhodnější k odfiltrování molekul z datové sady vcházející do HTS.

3.7.2 Spuštění testů na vzdálených počítačích

Popis situace Programátor LBVS metod vyvíjí novou metodu a chtěl by ji porovnat s ostatními dostupnými. Má svůj vlastní grid, na kterém si svou metodu chce testovat, jelikož ke spuštění potřebuje speciální knihovny. Platforma má vytvořený konfigurační soubor s definicí vzdálených počítačů dle subsekce 3.1 a instalovaný balíček SOAS. Programátor není jediný, kdo používá platformu, proto si chce zaregistrovat novou pracovní skupinu, kterou si oddělí své spuštěné testy.

1. Registrace

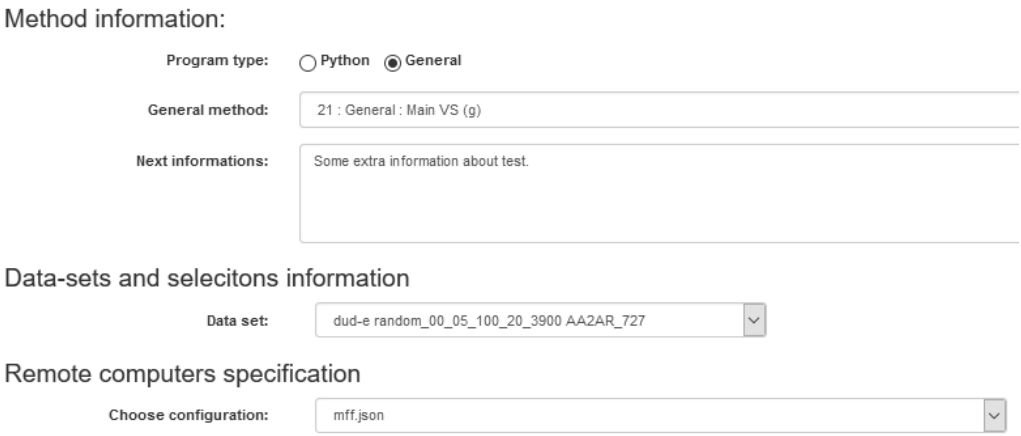
   Programátor si vytvoří novou pracovní skupinu na hlavní stránce *Programmer Number One* v levém formuláři.

2. Vložení metod

   Programátor vloží svou metodu a pokud nechce použít metody dodávané s platformou tak i všechny porovnávací pomocí menu *Method* položkou *Add*. Každá metoda musí být před vložením archivovaná ve formátu ZIP. Všechny potřebné soubory musí být umístěné v jedné složce, která je pojmenovaná názvem metody. V případě metody typu Python se musí vložit funkce *loading* a *scoring* zvlášť. Tlačítko *Add new method* odešle formulář a přidá metodu (její část).

3. Spuštění testů

   Programátor klikne v menu *Test -> Launch* na položku *Remote computers*. Svojí konfiguraci ke gridu vybere z nabídky *Choose configuration* v části *Remote computers specification*. Testy spustí pro všechny datové sady, na kterých bude chtít spustit porovnávací analýzu.

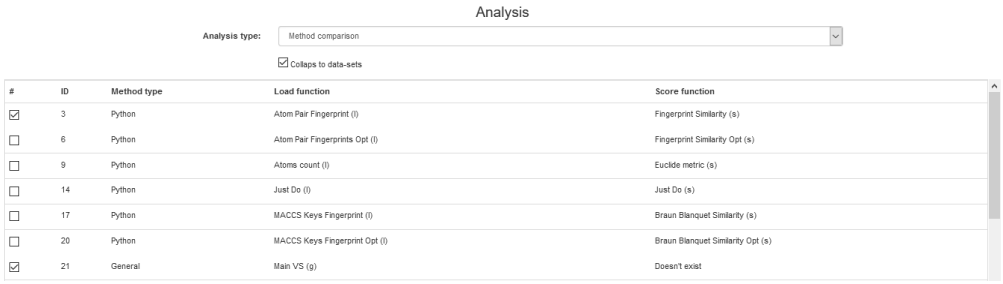


4. Kontrola testů

   Seznam nedoběhnutých a ještě nezkontrolovaných testů je udržován v přehledu na stránce *Test -> Running*.

5. Příprava výsledků k analýze

   Aby zjistil, jak se jeho metodě daří v konkurenci ostatních, nechá si udělat analýzu. Ná stránce, na kterou se dostane po kliknutí na odkaz *Analysis -> External analysis*, vybere z nabídky *Analysis type* položku *Method comparison*. Dále označí všechny metody, které chce do porovnání zařadit a zaškrtnutím check-boxu *Collapse to data-sets* sloučí v analýze výběry ze stejných datových sad. Jelikož průběh zpracování analýzy trvá delší dobu, je tento proces přesunut do externího skriptu. Po zmáčknutí tlačítka *Get input file* se objeví výzva k stáhnutí souboru *SOUB*, který přijde na vstup skriptu *analysis_scripts/external_analysis.py*.



6. Provedení analýzy

   Přesune se do složky *analysis_scripts*. Analýzu spustí příkazem

```
python external\_analysis.py -ip cestaKSouboruSOUB
[-op cestaKVystupnimuSouboru]
```

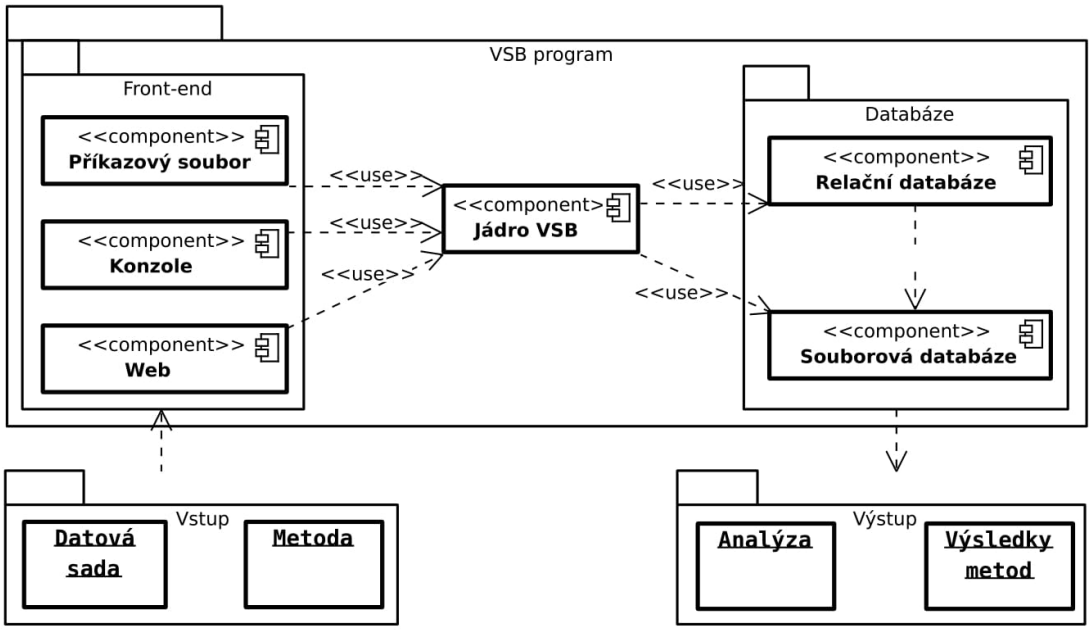
   Výchozí výstupní soubor je nastaven na aktuální složku se jménem souboru o.json.

7. Zobrazení výsledků

   Na webových stránkách po klinutí na položku *Show external analysis* v menu *Analysis*, vybere výstupní soubor z analýzy a klikne na tlačítko *Submit*, které nechá její výsledky graficky zobrazit. Popis analýzy je uveden v kapitole 3.5.

4. Programátorská dokumentace

Program se skládá z několika částí: dvou databází, jádra programu, různých typů rozhraní, vstupních a výstupních dat. Diagram 4.1 zobrazuje jejich vzájemnou interakci. Zdrojové soubory obsluhující databáze jsou integrovány v jádru. Zdrojové soubory grafického rozhraní jsou umístěny v samostatné složce *web_interface*. Mezi vstupní data se řadí metody a datové sady, mezi výstupní analýzy a výsledky metod.



Obrázek 4.1: VSB program

4.1 Datové sady a metody

Metody

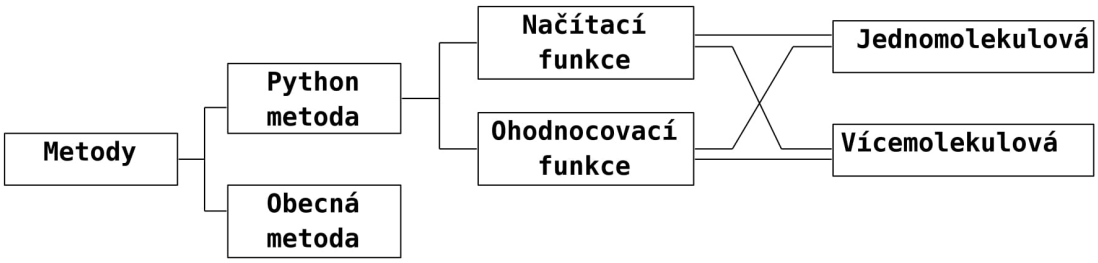
Diagram 4.2 ukazuje rozdělení vložitelných metod do platformy VSB.

Každá Python funkce (obecná metoda) je tvořena jedním adresářem nazvaným jejím jménem. Všechny zdrojové a aplikační závislosti je nutné mít umístěny v ní (pro případné spouštění metody na vzdálených počítačích). Hlavní spouštěcí soubor musí být pojmenován main_*.#, kde # je jakákoliv koncovka spustitelného souboru podporovaného testovacími stroji a * je substituce za jakékoliv podslovo splňující pravidla pojmenovávání souborů OS.

Program najde první soubor ve složce metody s názvem obsahujícím podslovo main a zkusí ho spustit. V případě neúspěchu program vypíše chybovou hlášku a ukončí se.

Z tohoto důvodu se nesmí ve složce metody nacházet žádný jiný soubor, v jehož názvu by se vyskytovalo podslovo main.

Metody přijímané programem jsou rozděleny do dvou druhů *Obecné* a *Python*. *Python* se dále dělí na jednomolekulové a vícemolekulové.



Obrázek 4.2: Rozdělení metod v programu

Python metody Metody psané pouze v jazyku Python a rozdělené na dvě funkce. První z nich *loading* zajišťuje nahrání molekul se souboru SDF a vypočítání reprezentací, *scoring* se stará o výpočet podobnostních výsledků.

Rozdělení Python metod na dvě funkce umožňuje zrychlit provádění výpočtů rozložením vykonávání úloh na více procesů a funkce mezi sebou různě kombinovat. Správnou kombinaci zajišťuje uživatel při zadávání testu, a tedy zodpovídá i za případné selhání programu při nesmyslné volbě.

Python metody lze rozdělit do dvou kategorií určených rozhraním, jenž implementují. Ke spuštění obou kategorií se využívá modul *skeleton*, do kterého se *loading* a *scoring* funkce jako vnější moduly dynamicky importují.

U jednomolekulové verze funkcí dochází automaticky k jejich spouštění pomocí více procesů.

Příklady jednomolekulových funkcí *test_func/Atom Pair Fingerprints Opt (l)* a *test_func/Braun Blanquet Similarity Opt (s)*.

Příklady vícemolekulových funkcí *test_func/Atom Pair Fingerprints (l)* a *test_func/Braun Blanquet Similarity (s)*.

Obecné metody Obecná metoda je jakýkoliv spustitelný soubor. Jelikož není znám typ souboru, dochází k jeho spuštění pomocí příkazové řádky. Uživatel zodpovídá za zadání validního a bezpečného příkazu, který je uveden v přidruženém souboru *method_configuration.json* v složce metody (klíč: execute_command).

Příkladem obecné metody je *test_func/Main VS (g)*.

Přidávání metod Program podporuje pouze přidávání obecných metod či Python funkcí, jejichž zdrojové a pomocné soubory jsou umístěny v jednom adresáři. Původní jméno metody (adresáře) je uloženo v relační databázi. Metoda je v souborové databázi uložena pod platformou generovaným jménem.

Funkcím *loading* a *scoring* je přiřazen právě jeden identifikátor a jsou ukládány zvlášť. Složení do celkové metody se děje virtuálně záznamem do relační databáze a vepsáním jejich ID do sloupce JIDS.

Obecné metodě odpovídá jeden identifikátor a má jeden záznam v relační databázi.

Datové sady

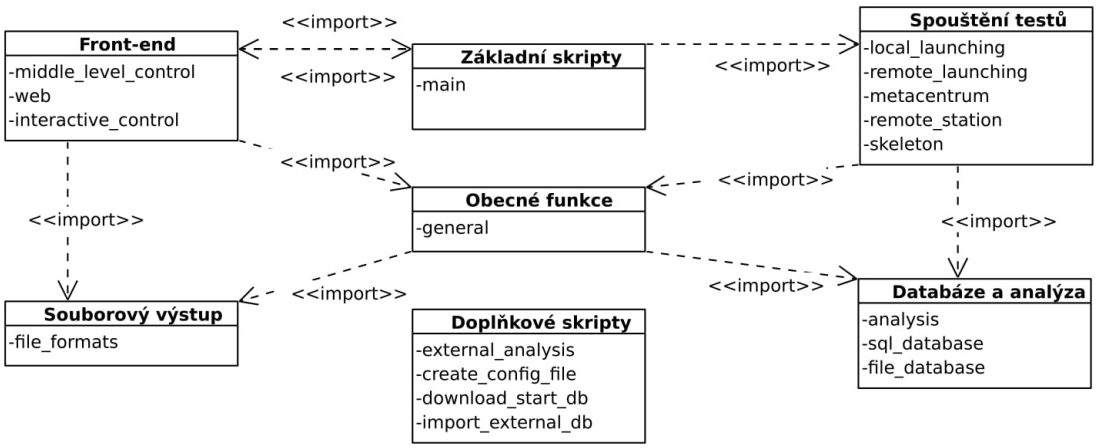
Aby se předešlo konfliktům ve jménech datových sad, ukládájí se přidané sady pod jménem ve formátu X_Y, kde X je původní jméno a Y značí unikátní číslo v databázi. Datová sada odpovídá adresáři, ve kterém je umístěn SDF soubor

s popisem použitých molekul a datový soubor s určením jejich aktivity. Dále se zde nachází adresář se složkami reprezentující výběry na datové sadě, které obsahují jejich definice.

Formát SDF je vybrán do platformy kvůli rozšířenosti a snadnému doplňování nadstandardních informací ke struktuře molekul. Tato reprezentace je stažitelná i z portálu PubChem.

4.2 Rozdělení programu a závislosti

Diagram 4.3 znázorňuje sloučení zdrojových souborů dle jejich funkcí do logických jednotek. V programu nejsou zdrojové soubory sloučeny do balíčků z důvodu množství křižných závislostí.



Obrázek 4.3: Závislostní diagram aplikačních zdrojových souborů

Zdrojový soubor *main.py* dle typu spuštění aplikace (-g,-a,-i) spustí příslušnou metodu zajišťující komunikaci s uživatelem, která se nachází v modulech *Front-end*. Při požadavku uživatele na získání (vložení) dat se vypíšou (zaznamenají) informace z (do) databáze. Přeje-li si uživatel spustit test, vrátí se řízení programu zpátky do modulu *main.py*, který požadavek zanalyzuje a předá řízení jednomu z modulů ze skupiny *Spouštění testů*. Tyto moduly využívají k zápisu informací do databáze funkce zapsané v modulu *general.py*.

4.3 Front-end

Web

Struktura Zdrojové kódy webových stránek jsou umístěny ve složce *web_interface*. V modulu *web.py* jsou zapsány funkce zajišťující chod webových stránek. Adresář *static* si udržuje soubory viditelné uživatelem a využité externí knihovny. Složka *templates* v sobě ukládá šablony webových stránek.

Webový server zajišťuje microframework Flask[1] napsaný v jazyku Python. Pro psaní šablon stránek se využívá knihovny Jinja2[2], která upravuje podmnožinu

[1] http://flask.pocoo.org
[2] http://jinja.pocoo.org/

jazyka Python. Používá se hierarchická struktura stránek, aby mohly od sebe navzájem dědit.

K zobrazení webových stránek, kromě domovské a dokumentace, je nutnost být přihlášený pod nějakou skupinou. V zdrojovém kódu jsou tyto stránky označené dekorátorem *@login_required*.

Ke stylizaci a dynamické funkčnosti webových stránek byla použita knihovna JQuery[3] a Bootstrap[4]. Vytváření grafů zajišťuje knihovna D3JS[5].

Většina funkčního kódu je programována na serverové straně. Z tohoto důvodu se při změně parametrů a kliknutí na tlačítko typu *submit* odešlou údaje zpět na server a vygeneruje se nová stránka.

Dočasné soubory se ukládají ve složce *web_interface/temp* či v případě obrázků určených k zobrazení na webu v *web_interface/static/img*. Po ukončení běhu serveru dojde k vymazání těchto složek.

Web.py importuje *main.py*, aby komunikace s databázemi byla co nejrychlejší. Příkazy na straně VSB aplikace vykonává modul *middle_level_control.py*.

Spouštění testu na webových stránkách vytváří ze vstupních parametrů příkazový soubor. S tímto vstupním souborem je aplikace VSB spuštěna pomocí modulu multiprocessing. Proces je uložen s dalšími údaji do datového pole, aby bylo možné poskytovat aktuální informace o probíhajících testech.

Konzole

Konzole je zakomponována do VSB aplikace pomocí modulu *interactive_control.py*, který importuje moduly pro obsluhu databází. To umožňuje přistupovat přímo do nich při získávání a ukládání dat. Spouštění testů probíhá přes příkazový soubor vygenerovaný po zadání jejich detailů.

Souborový přístup

Všechny operace, které lze dělat ve webovém rozhraní, lze i v souborovém přístupu, jelikož jsou obě dvě rozhraní obsluhována stejným modulem *middle_level_control.py*. Tento modul importuje všechny důležité moduly pro běh testů a práci s databázemi. Hlavní funkcí v něm je *database_actions*, jejímž prvním parametrem je slovník obsahující zadání požadavku a *True* v druhém parametr značí, že slovník obsahuje klíč *output_file*, do nějž se ukládá cesta k výstupnímu souboru.

4.4 Spouštění testů

Každou z následujících možností zajišťuje jiný modul a tvoří ji rozdílný algoritmus.

Lokální počítač Lokální spouštění zajišťuje modul *local_launching*. Sekvenční postup spustí požadovanou metodu N-krát za sebou. Paralelní spouštění využívá

[3] https://jquery.com/
[4] http://getbootstrap.com/
[5] https://d3js.org/

multiprocessing modul a rozděluje výpočty do několika procesů. Jejich maximální počet je omezen počtem jader procesoru. Vlákna nelze v jazyku Python použít, protože nefungují zcela jako nezávislé jednotky kvůli global interpreter lock (GIL), jenž nedovolí vykonávat dva stejné kusy kódu ve stejný čas.

Vzdálené počítače Data o počítačích se čtou ze souboru ve formátu *examples/pcToUse.json*. Mezi podporovanými systémy jsou Linux a Windows. K vzdáleným počítačům určených pomocí adres uvedených u klíčů *hostname* se aplikace připojuje skrz knihovnu pySFTP. Na každý počítač je přenesen výběr s SDF souborem, konfigurační soubor, modul *remote_machine* a metoda. Dle jejího typu může být dále přenesen soubor s upravenými vstupními daty a modul *skeleton* v případě typu Python. Na vzdáleném počítači se vystaví adresářová struktura *VSB_folder/method_id(s)/test_X_Y/soubory*, kde X značí jméno datové sady a Y jméno výběru.

Modul *remote_station* během běhu vytvoří dva soubory. Obsah *State* nabývá dvou logických hodnot a určuje stav vykonávání testu (ukončen/vykonáván). Soubor *success* má uloženou hodnotu True/False dle úspěchu provedení testu.

Během testu se přesměrují všechny výstupy směřované do konzole do souborů stderr.txt a stdout.txt, jenž jsou po skončení testu přeneseny zpět na lokální počítač.

Po ukončení vykonávání metody na vzdáleném počítači se výsledky přenesou zpět na lokální počítač, vzdálená složka se vymaže a v případě potřeby se spustí nový test na výberu.

Výsledky se na hlavním počítači zanalyzují a zaznamenají do databáze.

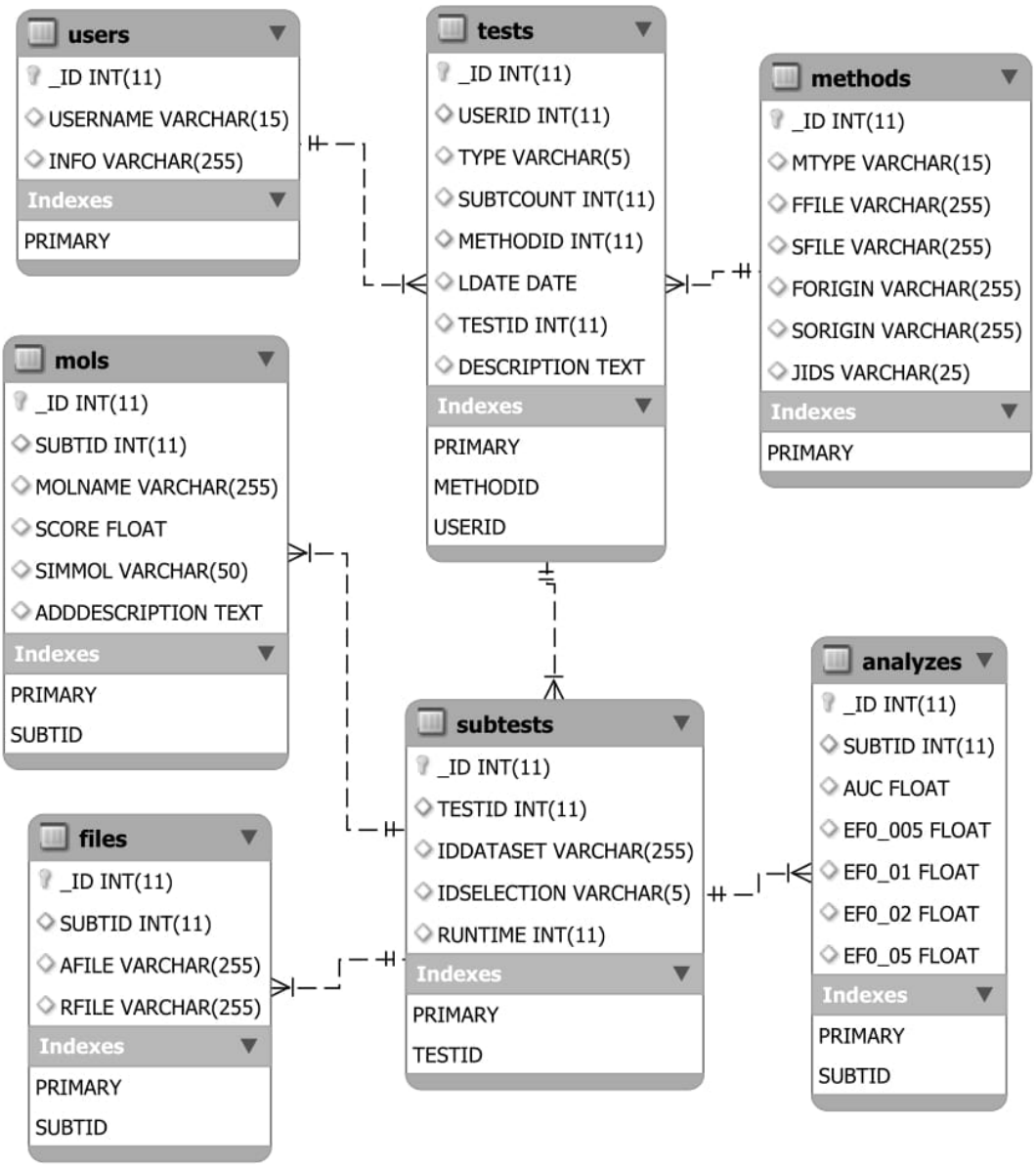
Metacentrum Zahájení testů na infrastruktuře Metacentrum probíhá registrací úlohy v systému PBS. Úkol je registrován spuštěním příkazu *qsub* na příkazové řádce tamního řídícího počítače s označením cesty ke skriptu, ve kterém jsou zapsány úlohy k vykonání, a požadavky na žádaný počítač. Jelikož úložný prostor je mezi počítači sdílený, lze přenést na diskové pole všechny soubory najednou. Adresářová struktura je stejná jako u spouštění na vzdálených počítačích. Jakmile je úloha zaregistrována, je jí přidělen identifikační kód, přes který se program pomocí příkazu *qstat* dotazuje, zda je již úloha dokončena. Následný postup stažení a zpracování výsledků využívá funkce z modulu pro vzdálené počítače.

4.5 Datový model

Program obsahuje dvě databáze souborovou a relační. Do první z nich se ukládají uživatelské metody, datové sady, skóre jednotlivých testů na výběrech a jejich výstupy. Do relační databáze se ukládají všechny informace o provedených testech vygenerované během běhu platformy např. počet vykonaných testů na výběrech, názvy datových sad či číselné analýzy výsledků.

Relační databáze si udržuje informaci o množství metod uložených v jejím souborovém protějšku. Při dotazu na jejich jména se neprohledává souborová DB, nýbrž je dotaz veden na výpis metod uvedených v relační DB.

SQL databáze



Obrázek 4.4: ER diagram SQL databáze

Struktura Relační databáze (RDB) se skládá celkem ze sedmi tabulek uspořádaných dle diagramu 4.4.

V tabulce metody sloupce MTYPE odpovídají typům metod (py, general), FFILE/SFILE identifikátorům přiřazeným metodám v souborové databázi, FORIGIN/SORIGIN původním jménům. Sloupec JIDS ukládá SQL identifikátory ve formě ID1#ID2 určující SQL ID *loading* a *scoring* funkcí, ze kterých je metoda typu *py* vytvořena. V případě, že do databáze ukládáme pouze *loading* nebo *scoring* funkci, příp. general metodu nevyužitá pole vyplňujeme řetězcem *Empty*.

Sloupce v tabulce testů tvoří kromě referenčních klíčů na použitou metodu a spouštějícího uživatele i typ testu (možnosti, které lze použít jsou uloženy ve

třídě *DatabaseHelper.Test_types* v modulu *sql_database.py*), počet zadaných subtestů (SUBCOUNT), den, kdy byl test spuštěn (LDATE), ID přiřazené testu v souborové databázi (TESTID).

Subtest zastupuje spuštění metody na jednom výběru.

Files obsahují cesty k souborům s výstupem analýzy (AFILE) a metody (RFILE).

Sloupec SIMMOL v tabulce mols ukládá pro danou molekulu metodou označenou known active a ADDDESCRIPTION si pamatuje dodatečný popis zanechaný metodou.

Typy databáze Program poskytuje možnost připojení se ke dvěma typům databází. Prvním z nich je SQLite3, která je poskytována jako modul v stadardní knihovně jazyka Python. Tato databáze ukládá své data do jediného souboru *benchmark.db*, jehož umístění je ve složce *data*. Druhá možnost se složitější instalací je samostatný MySQL server. MySQL server se vyznačuje rychlejším zpracováním a chytřejším uložením dat. Ke správné funkci je vyžadován ovladač PyMySQL zařizující komunikační agendu se serverem.

Souborová databáze

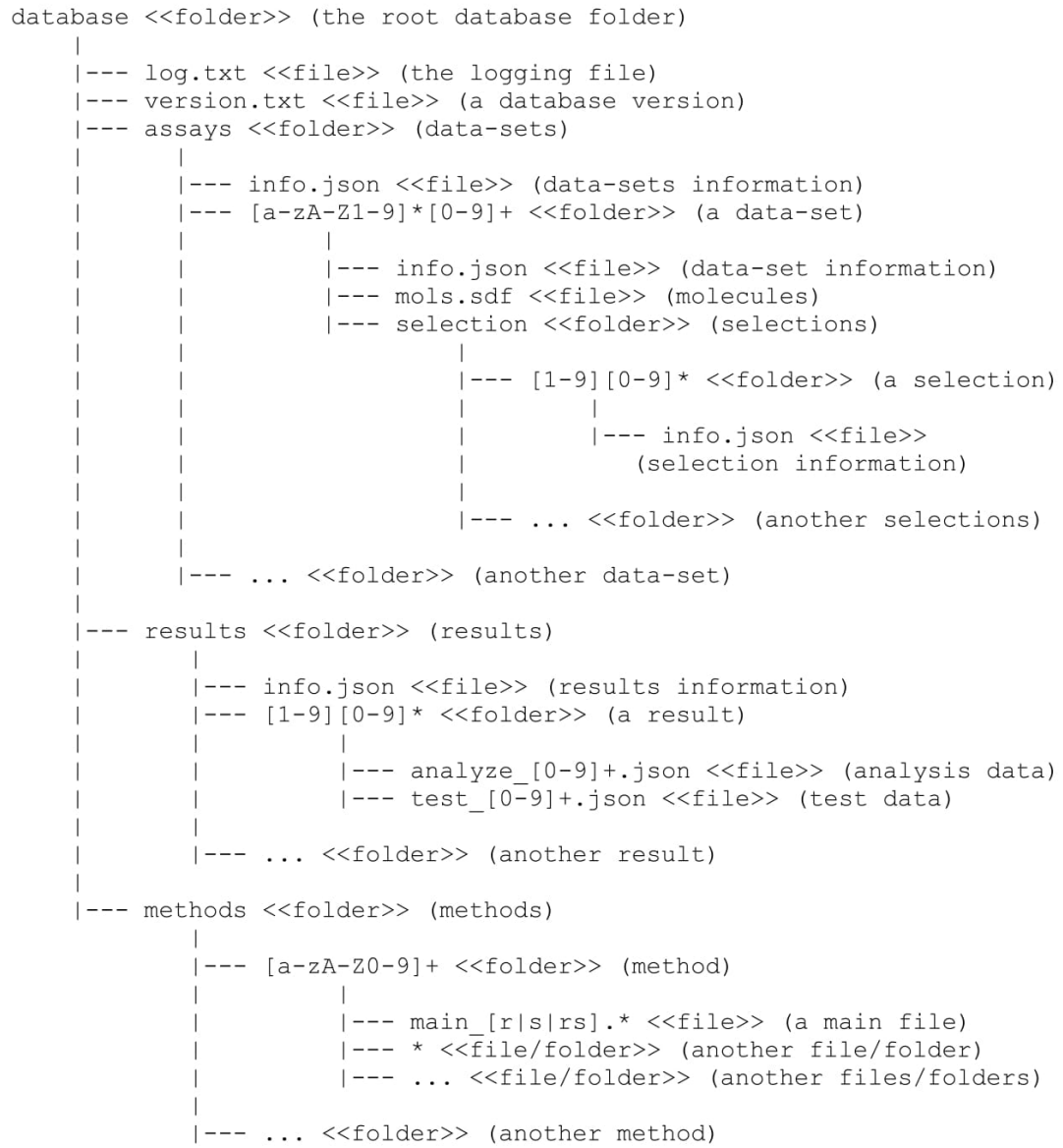
Struktura Diagram 4.5 ukazuje adresářové rozdělení tvořící souborovou databázi (SDB). Jména složek a souborů musí splňovat zápis pomocí regulárních výrazů u nich uvedených a zároveň omezení dané operačním systémem, na kterém aplikace běží. Všechny datové soubory kromě obsahujicích popis molekul, verzi databáze a logovací data jsou ukládány ve formátu json.

Ve složce souborové databáze *database* se nachází dva textové soubory. Logovací soubor v sobě ukládá záznamy o provedených operacích v RDB i SDB. Druhý soubor má v sobě uloženou verzi a čas poslední aktualizace demostračního balíčku datových sad či vzniku složky *database*.

Ukládání výsledků testů Po každém provedeném testu se ukládájí výsledky ohodnocení metodou a výsledná analýza provedená na nich do složky se jménem značícím pořadové číslo testu, které je dané ID testu v RDB. Složka přiřazená testu se zakládá při každém jeho spuštění tzn., že pokud test z nějakých důvodů nevyšel, zůstane adresář prázdný. Počty souborů *analyze_X* a *test_X* se v něm vyskytují maximálně tolikrát, kolik subtestů bylo provedeno. Dále kromě výše zmíněných dat může složka obsahovat i soubory s chybovým a standardním výstupem programu či metody.

Operace s databázemi

Jelikož lze mít instalovaných více instancí aplikace najednou, nabízí platforma slučování databází. Při něm dochází k přesunům záznamů z SQL i souborové databáze. Aby se předešlo velkým přesunům dat, nekopíruji se vzájemně datové sady. Dochází ke kopírování metod, tzn. pokud se metoda již v instanci, do níž se kopíruje nachází, je vložena do seznamu metod ještě jednou. Výsledný objekt vytvořený funkcí pro přesun se skládá ze složky *move*, která obsahuje jeden soubor *info.json*, v němž jsou uloženy všechny informace, jež se následně vloží do RDB,



Obrázek 4.5: Diagram struktury souborové databáze

a složky zastupující testy, které obsahují soubory patřící k testům (tj. všechny výstupy + metody). Uplatněním inverzního algoritmu pro získání dat z databáze, se data do cílové instance vloží.

4.6 Výsledky metod a analýza dat

Výsledky metod

Dle typu spuštěné metody se mohou objevit dva různé postupy zápisu výsledku testů. V případě obecné metody dochází k zápsání ohodnocení molekul do výstupního souboru samotnou metodou. V případě Python typu je vrácen výsledek jako návratová hodnota, která je zapsána platformou do tohoto formátu souboru. Soubor slouží k archivaci výsledků v souborové databázi.

Soubory jsou typu json. Kromě dvou povinných klíčů *name* (jméno molekuly) a *similarity* (přidělené skóre) pro každou molekulu jsou další informace dobrovolné. Program je schopen přečíst *score_molecule* (molekula, která byla testované molekule nejpodobnější) a *description* (dodatečná krátká informace, která je k molekule přiřazená). Metoda není omezena žádnými restrikcemi na dodatečný obsah souboru, tudíž si sama může další informace přidávat.

Analýza

Analýza je vyhodnocována knihovnou RDKit na výsledcích provedeného testu. Zpracování zajišťuje funkce *analyze* v modulu *src/analysis.py*.

Jelikož některé analýzy trvají delší dobu, jsou přesunuty do modulu *external_analysis.py* v složce *analysis_scripts*. Modul dostane na vstupu *json* soubor, ve kterém je jméno analýzy a její vstupy. Dle jména se určí analýza, přečtou se vstupy a spustí se hlavní funkce reprezentující analýzu.

Klastrování je prováděno funkcí *make_clustering_on_vectors*, která používá dle potřeby *Euklidovou* a *Hammingovou vzdálenost*.

5. Experiment

Pro demonstraci funkčnosti a použitelnosti platformy jsem provedl dva experimenty.

5.1 Ověření standardních metod

V článku *Benchmarking Platform for Ligand-Based Virtual Screening* [25] publikovali autoři svou platfromu na testování LBVS metod.

K testování efektivity vybraných metod (*Topological torsion fingerprints* (TT), *Morgan fingerprints* s parametry 1 a 2 (M1, FM1, M2, FM2), *Atom pairs fingerprints* (AP) a *MACCS keys fingerprints* (MACCS) s podobnostní metrikou *Tanimoto similarity*) použili kolekci 00 z balíku datových sad SOAS. Kolekce rozděluje datové sady do složitostních kategorií dle získaného AUC od referenčních metod. Během pokusu bylo ukázáno, že existuje tendence klesání úspěšnosti testovaných metod se zvyšující se složitostí datových sad.

V prvním experimentu budu replikovat publikací popsaný experiment a tím se pokusím ověřit jeho správnost a reprodukovatelnost. Při pokusu budou využity stejné datové sady a metody, které byly použity v původním pokusu. K výpočtu AUC hodnot nebude využito w-AUC (při stejných skóre molekul se upřednostní neaktivní molekuly), ale seřazení stejně ohodnocených molekul bude náhodné. Toto řazení sice může vést k lepším hodnotám AUC, ale mělo by zanechat pozorovaný trend.

Domnívám se, že za daných předpokladů budou grafy podobné a bude existovat sestupná tendence pro všechny testované metody.

Výsledky pokusu

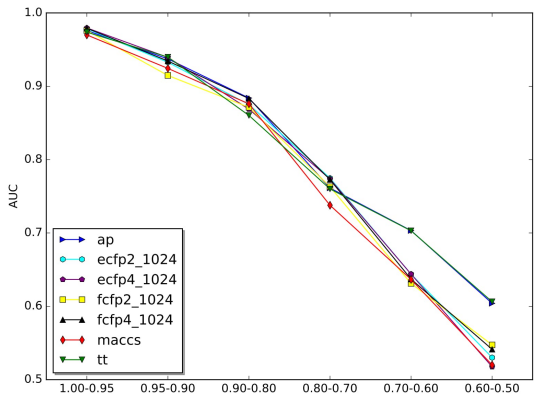
Graf 5.1 ukazuje výsledky získané při pokusu v publikaci [25] a 5.2 hodnoty vypočítané na platformě VSB. Porovnáním obou grafů lze vidět, že všechny testované metody sestupnou tendenci splňují a grafy se ve větší časti shodují. Rozdílné výsledky dostaváme pouze pro metodu AP, která v testech na VSB dopadla o něco hůře než v původním pokusu. Tato změna může být způsobena použitím rozdílného seřazení molekul.

5.2 Ověření BG metody

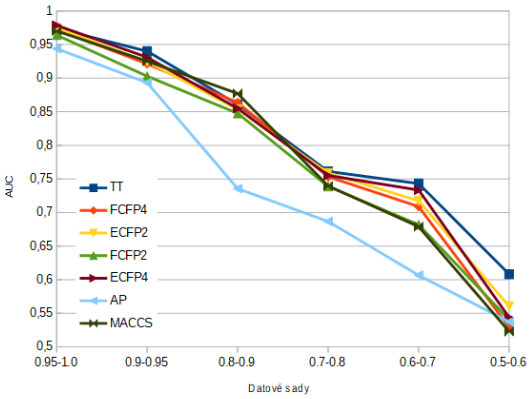
Ve svém článku publikovali pánové Bender, A. & Glen, R. C. [22] novou metodu (BG metoda) založenou na porovnávání počtu atomů v molekulách. Metoda využívá jako fingerprinty *BG fingerprints* a pro výpočet podobnostního skóre *Eukleidovu metriku*. Efektivita metody byla ověřena na jimi získaných datových sadách a porovnávána se standardními metodami.

Výsledek jejich pokusu ukázal, že BG metoda se dokáže vyrovnat výkonnosti standardních metod.

Abych tuto skutečnost ověřil, spustil jsem mnou poskládáné metody a BG metodu na datových sadách *COX2*, *DHFR*, *EGFR*, *FXA*, *P38*, *PDGFRB* a *SRC*



Obrázek 5.1: Průměrné AUC - publikace



Obrázek 5.2: Průměrné AUC - VSB

ze skupiny *10.1021%2Fjm300687e* a rozdělení *random_01_50_100_20_4900_M* z balíku datových sad SOAS.

Načítávací a podobnostní funkce u pokusných metod jsou voleny tak, aby zachycovaly širší diverzitu. Všechny části jsou ovšem standardně dostupné v knihovně RDkit.

Do porovnávacích metod jsou zařazeny *Atompairs fingerprints + Cosine similarity* (APF-CS), *Atoms count + Euclide metric* (AC-EM), *MACCS keys fingerprints + Braun Blanquet similarity* (MACCS-BBS), *Morgan fingerprints + Tanimoto similarity* (MF-TS), *Topological torsion fingerprints + Dice similarity* (TTF-DS).

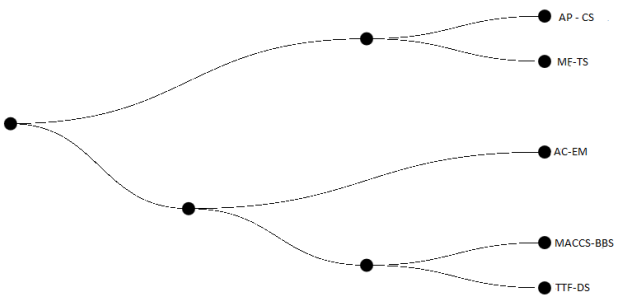
Jelikož *BG metoda* používá k porovnávání molekul a výpočtu skóre prosté principy a přesto její výkonnost byla v publikovaných testech dobrá, předpokládám, že to mohlo být způsobeno jejich jednoduchostí. Očkávám, že při použití obtížnějších datových sad, bude její efektivita horší než stadandardních metod.

Výsledky experimentu

Na grafu průměrných hodnot AUC na testovaných datových sadách 5.4 lze zpozorovat, že zkoumáná metoda nedosáhla dobrých výsledků u žádné datové sady. U všech datových sad si standardní metody vedly podstatně lépe. Tabulka 5.1 zobrazuje průměrné výsledky metriky EF. V úvahu bylo vzato prvních 0,005% z průměrně 4 750 testovacích molekul na jeden výběr. V testovaných molekulách se průměrně vyskytovalo 20 ligandů. Červeně zvýrazněné buňky značí nejnižší hodnotu a zelené naopak nejvyšší. Je patrné, že hodnoty z tabulky se podobají výsledkům z grafu, přičemž metoda *AC-EM* získala nejhorší skóre u všech případů.

Podle grafu i tabulky si ze standardních metod na testovaných datových sadách vedla nejhůře metoda *MACCS - BBS*, která v metrice EF jako jediná nezískala ani jednou nejvyšší skóre. Ostatní metody měly podobné výkonosti.

Na otestovaných metodách bylo provedeno několik různých klastrování. V klastru založeném na podobnosti řetězců hodnot tvořených základním vyhodnocením výsledků metod (AUC, EF) byla *BG metoda* nejdříve sloučena s metodou *MACCS-BBS*, poté s *TTF-DS* a následně s ostatními. U porovnávání známých aktivních (*cluster on fingerprints created by known active on same places* 3.5) byla

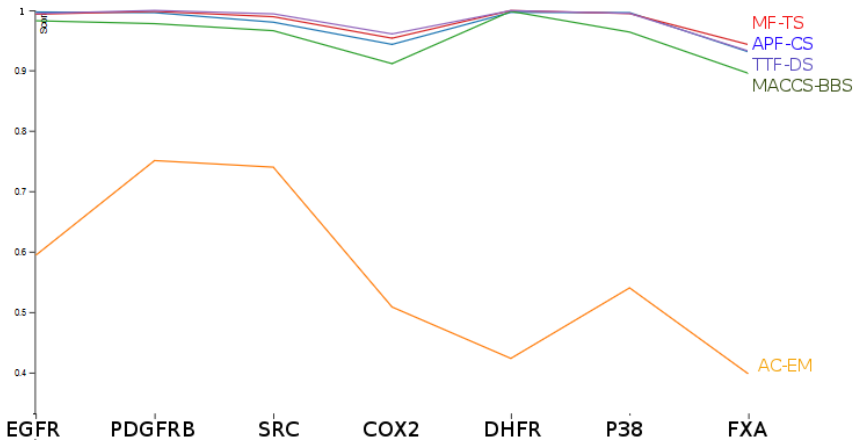


Obrázek 5.3: Cluster *Collapse fingerprints*

AC-EM metoda u většiny případů přidána do společného shluku až v poslední iteraci. Při ohledu na nízkou výkonnost lze předpokládat, že k ohodnocování molekul vybírala méně vhodnější *known actives* než ostatní metody. U klastrů vytvořených pomocí řetězců zakládajících se na relativních rozdílech skóre a odstupech aktivních molekul byla veliká diverzita, tudíž všechny testované metody jsou v těchto hlediscích zcela odlišné. Při testování vzdáleností řetězců vyrobených postupem *collapse fingerprints* 3.5 s parametrem velikosti okna 10 molekul a počtem oken 40 vytvořila *AC-EM* metoda ve všech případech klastr zobrazený na obrázku 5.3. Porovnáním informací získaných ze všech klastrů a analýz vyplývá, že *BG metoda* je ostatním nejméně podobná.

Tabulka 5.1: EF 0.005 hodnoty

Datové sady	APF-FS	AC-EM	MACCS-BBS	MF-TS	TTF-DS
EGFR	19.54155	6.90210	18.30841	19.42633	19.32645
PDGFRB	19.85627	10.8289	18.64319	19.75656	19.75656
SRC	18.84357	7.07921	17.94628	19.44173	19.34204
COX2	16.86771	2.59501	16.06967	18.06564	18.06541
DHFR	19.66198	1.89562	19.66242	19.76190	19.86155
P38	19.16003	2.02345	16.41704	19.66983	19.57001
FXA	16.46143	1.49522	14.85892	16.76060	16.55991



Obrázek 5.4: Průměrné hodnoty AUC na všech molekulách (Y osa zobrazuje AUC hodnoty 0.4 až 1, X osa udává datové sady)

Závěr

Cílem bakalářské práce bylo navrhnout a implementovat platformu pro testovaní efektivity ligand-based virtual screening metod. Platforma měla poskytnout jednoduchou manipulaci s daty a možnosti analýzy výsledků metod.

Vzhledem k existenci mnoha LBVS metod a testovacích dat, jsem implementoval podporu pro paralelní běh, která umožňuje metody vyhodnocovat v přijatelném čase. Pro paralelní běh jsou nabízeny možnosti spouštění testů na infrastruktuře Metacentrum a vlastním gridu. Pro komunikaci s platformou slouží webové stránky, konzole a souborové příkazy. Práce s webovými stránkami byla popsána na případech užití v uživatelské dokumentaci, kde jsou též vysvětleny implementované analýzy výsledků metod.

Možnosti využití platformy jsou demonstrovány na dvou příkladech. Prvním je replikace publikovaného experimentu a druhým ověření *BG metody* na nových datech.

Reprodukcí prvního pokusu se potvrdila skutečnost, že existuje trend klesání úspěšnosti testovaných metod se zvyšující se složitostí datových sad.

Druhý experiment ukázal, že *BG metoda* dosahuje slabých výsledků na námi použitých datových sadách. Dobré hodnocení, které získala v testech u autorů, je tedy možné přisoudit jejich datovým sadám.

Provedené experimenty demonstrují použitelnost naprogramované platformy. Jejich výsledky ukazují užitečnost její existence při ověřování publikovaných výsledků.

Literatura

[1] The Editors of Encyclopedia Britannica. Pepsin. 2015.

[2] John P. Overington, Bissan Al-Lazikani, and Andrew L. Hopkins. How many drug targets are there? *Nature*, 2006.

[3] National Cancer Institute. Nci dictionary of cancer terms.

[4] Ph.D. Mindy I. Davis. A faster kayak: Mass spectroscopy for multiplexed kinase assays. *Genetic Engineering & Biotechnology News*, 2012.

[5] Nuffield Department of Medicine and University of Oxford. High throughput screening - an introduction.

[6] the University of Wisconsin-Madison UW Health. Uwccc small molecule screening facility. *uwhealth.org*.

[7] Novartis. Robots speed the pace of modern drug discovery. 2014.

[8] Sereina Riniker and Gregory A Landrum. Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of Cheminformatics*, 2013.

[9] Joseph Castro. How do enzymes work? 2014.

[10] Milada Teplá. Přírodní látky - enzymy. *studiumbiochemie.cz*.

[11] Singer Instruments. What is high throughput screening? *singerinstruments.com*.

[12] Brian K. Shoichet. Virtual screening of chemical libraries. *Nature*, 2004.

[13] Nikolas Fechner-Andreas Zell Andreas Jahn, Georg Hinselmann. Optimal assignment methods for ligand-based virtual screening. *Journal of Cheminformatics*, 2009.

[14] Robert P. Sheridan and Simon K. Kearsley. Why do we need so many chemical similarity search methods? *Drug Discovery Today*, 2002.

[15] J.H. Schurr. The coding of three-dimensional structure of molecules by molecular transforms and its application to structure-spectra correlations and studies of biological activity. *Journal of chemical information and modeling*, 1996.

[16] Claire M.R. Ginn, Peter Willett, and John Bradshaw. Combination of molecular similarity measures using data fusion. *Perspectives in Drug Discovery and Design*, 2000.

[17] V.J. Gillet Andrew R. Leach. *An Introduction to Chemoinformatics*. Springer Science & Business Media, 2003.

[18] John D. MacCuish and and Mitch Chapman Norah E. MacCuish. The fingerprint module in the mesa suite. *Mesaac*, 2010.

[19] Cambridge MedChem Consulting. Directed and virtual screening. *Cambridge MedChem Consulting*, 2012.

[20] Christoph Sotriffer. *Virtual Screening: Principles, Challenges, and Practical Guidelines*. John Wiley & Sons, 2011.

[21] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and modeling*, 2002.

[22] Andreas Bender and Robert C. Glen. A discussion of measures of enrichment in virtual screening: Comparing the information content of descriptors with increasing levels of sophistication. *J. Chem. Inf. Model*, 2005.

[23] Charly Empereur-mot, Hélne Guillemain, Aurélien Latouche, Jean-François Zagury, Vivian Viallon, and Matthieu Montes. Predictiveness curves in virtual screening. 2015.

[24] Wei Zhao, Kirk E Hevener, Stephen W White, Richard E Lee, and James M Boyett. A statistical framework to evaluate virtual screening. *BMC Bioinformatics*, 2009.

[25] Petr Škoda and David Hoksza. Benchmarking platform for ligand-based virtual screening. 2016.