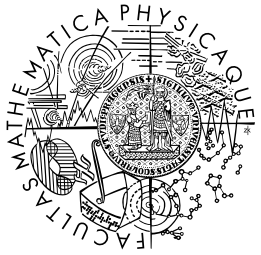


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Aneta Šťastná

Grafová reprezentace molekul a její využití ve virtuálním screeningu

Katedra softwarového inženýrství

Vedoucí bakalářské práce:	RNDr. David Hoksza, Ph.D.
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

Ráda bych poděkovala všem, kdo mě podpořili při psaní této práce, zejména však mým rodičům za materiální podporu, Bc. Tomáši Pokornému dodání motivace a za technickou podporu a také Bc. Ondřeji Mičkovi za dočasné zastání mé úlohy při organizování korespondenčního semináře M&M. Chtěla bych poděkovat svému vedoucímu, RNDr. Davidu Hokszovi, Ph.D., za čas věnovaný konzultacím a za nabídku zajímavého tématu. Mgr. Petru Škodovi děkuji za poskytnutí výsledků stávajících metod pro srovnání.

Nakonec bych chtěla poděkovat za výpočetní zdroje, které byly poskytnuty Ministerstvem školství, mládeže a tělovýchovy České Repupliky v rámci projektů CESNET (projekt LM2015042) a CERIT Scientific Cloud (projekt LM2015085), spadajících do probramu Projekty velkých infrastruktur pro VaVaI.

Název práce: Grafová reprezentace molekul a její využití ve virtuálním screeningu

Autor: Aneta Šťastná

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. David Hoksza, Ph.D., Katedra softwarového inženýrství

Abstrakt: Grafy jsou přirozeným způsobem reprezentace molekul. Přesto se grafové reprezentace a algoritmy běžně nepoužívají ke zjišťování podobnosti molekul ve virtuálním screeningu. V této práci testujeme grafové přístupy ve virtuálním screeningu založeném na ligandech. Podobnost molekulových grafů počítáme pomocí největšího společného podgrafu a grafové editační vzdálenosti. Používáme implementace `fmcs` z chemoinformatické knihovny RDKit pro určení největšího společného podgrafu a GraphMatchingToolkit od K. Riesena pro určení grafové editační vzdálenosti. Nalezli jsme vhodné kombinace parametrů pro aplikaci ve virtuálním screeningu. Výsledky ukazují, že grafové přístupy jsou kvalitativně srovnatelné se stávajícími metodami.

Klíčová slova: graf virtuální screening podobnost

Title: Graph-based molecular representation and its utilization in virtual screening

Author: Aneta Šťastná

Department: Department of Software Engineering

Supervisor: RNDr. David Hoksza, Ph.D., Department of Software Engineering

Abstract: Graphs are natural way of representing molecules. However, graph representations and algorithms are not being used for finding similarity of molecules in virtual screening. In this work we test the graph-based methods in ligand-based virtual screening. The similarity of molecular graphs is determined by maximum common subgraph and graph edit distance. We use implementations `fmcs` from chemoinformatic library RDKit for maximum common subgraph and GraphMatchingToolkit from K.Riesen to determine graph edit distance. We have found suitable combinations of parameters for aplication in ligand-based virtual screening. The results suggest that performance of graph based methods is comparable to the state-of-the-art methods.

Keywords: graph virtual screening similarity

Obsah

Úvod

Uvedení do problematiky

V posledních několika desetiletích se informatika začíná prolínat s mnoha ostatními obory, včetně chemie. Nezřídka tak vznikají zcela nové obory, jejichž smyslem je využít technického pokroku pro zlepšení či zrychlení výzkumu v jiných oblastech. Jedním z těchto oborů je chemoinformatika. Mezi její hlavní cíle patří hledání vhodných způsobů reprezentace molekul v počítači, ukládání molekul do databází za účelem efektivního vyhledávání podle zadaných parametrů a provádění různorodých výpočtů nad molekulami.[29]

Již první ze zmíněných cílů není tak snadný, jak by se mohlo zdát. Na reprezentaci molekul existuje mnoho často protichůdných požadavků, zmiňme alespoň možnost uložit prostorovou orientaci molekuly, volné elektrony či elektronové páry, aromaticitu vazeb, počet neutronů, koordinační vazby a 2D souřadnice. Zároveň je často požadováno, aby byl formát prostorově efektivní, neobsahoval atomy vodíku (jejichž přítomnost se často považuje za implicitní) a byl lidsky čitelný. Právě kvůli protichůdným a často poměrně komplexním požadavkům na formát uložení molekul a také kvůli neexistenci normy bylo vyvinuto mnoho rozličných formátů. Většinou formát klade důraz na jeden až dva požadavky a zbytek nesplňuje vůbec či jen omezeně. Vzhledem k rozmanité povaze jednotlivých formátů a chybějícím informacím není vždy možné provést automatickou konverzi mezi formáty. Z volně dostupných softwarových nástrojů podporujících velké množství formátů a jejich konverzi patří OpenBabel[1]. Více o dostupném chemoinformatickém softwaru a jednotlivých formátech viz kapitola 5. Rozličnost formátů vyplývá také z mnoha různorodých úkolů, kterými se chemoinformatika zabývá. Jedním z nich je i téma této práce.

V této práci se budeme zabývat virtuálním screeningem, který se používá zejména v procesu hledání nových léčiv.

Hlavním cílem **virtuálního screeningu** je prohledat databázi známých molekul a setřídit ji podle schopnosti molekuly interagovat s danou bílkovinou či enzymem. Na základě tohoto setřízení jsou vybrány molekuly s větší šancí na projevení biologické aktivity. [19] Potřeba udělat výběr nejperspektivnějších sloučenin vyplývá z počtu známých sloučenin, který se pohybuje v řádech desítek miliónů. Například veřejná databáze známých sloučenin PubChem[46] obsahuje 90 miliónů záznamů[2] a stále se rozrůstá. Virtuální screening se postupně stává důležitým mezikrokem při vývoji nových léčiv, protože je díky němu možné zavrhnout většinu neperspektivních sloučenin a otestovat v laboratoři jen ty, které budou mít potřebné vlastnosti s větší pravděpodobností.

Christopher A. Lipinski a kol. určili pravidla, kterými se vyznačuje většina látek schopných se v těle šířit, být absorbovány, metabolizovány a vylučovány.[50] Tím značně zúžili množství molekul přicházejících při vývoji léčiv v úvahu. Samotné splnění pravidel však nezaručuje biologickou aktivitu. Mezi pravidla patří omezení počtu určitých skupin a také omezení na celkovou molekulovou hmotnost.

[1] http://openbabel.org/

[2] Údaj z dubna 2017 pro PubChem Compound obsahující pouze unikátní sloučeniny.

Z pravidel tudíž vyplývá, že bioaktivní molekuly se vyznačují poměrně malou velikostí, typicky v řádech menších desítek atomů. Jedním z biologických důvodů tohoto jevu je například schopnost látky se rozpouštět a prostupovat membránami. Tato schopnost je přitom klíčová pro dopravu léčiva na správné místo v organismu. Díky tomuto omezení na velikost potenciálních léčiv je také možné při virtuálním screeningu testovat i algoritmy, které mají vyšší výpočetní složitost vzhledem k počtu atomů a vazeb.

Schopnost molekuly interagovat s danou makromolekulou se odvíjí od schopnosti se navázat do jejího **aktivního místa**. Po navázání molekuly do aktivního místa dojde k inhibici nebo naopak aktivaci dané makromolekuly. Tento důsledek bývá způsobený většinou čistě změnou tvaru aktivního místa po navázání molekuly, přestože vliv mohou mít i další efekty.

Proto je jedním ze způsobů posuzování aktivity molekul výpočet jejich přesného tvaru. Následně se vyhodnocuje míra komplementarity tvaru molekuly a aktivního místa. Nejčastější metoda výpočtu je tzv. **dokování molekul**, kdy je snahou najít přesný způsob zapadnutí molekuly do aktivního místa. Tento přístup bývá označován jako **virtuální screening založený na struktuře** *(structure-based virtual screening, SBVS)*.[47] Velkou výhodou tohoto přístupu je schopnost odhalit i úplně nové typy molekul zapadajících do aktivního místa. Mezi hlavní nevýhody SBVS patří výpočetní náročnost a potřeba znát strukturu cíle.

Kvůli tomu se vyvíjí i jiné přístupy k virtuálnímu screeningu. Přístup, který se používá v této práci, je založený na znalosti několika molekul, které se do aktivního místa dokážou navázat a jejich srovnávání s molekulami v databázi. Výše zmíněný přístup, označovaný jako **ligand-based** *(ligand-based virtual screening, LBVS)*[18], je obecně založený na molekulární podobnosti. Vychází z **principu podobnosti**, tedy z hypotézy, že podobné molekuly mají podobné chemické a biologické vlastnosti. Tento princip začali používat mezi prvními Adamson a Bush.[8, 9] Princip podobnosti dává poměrně dobrý smysl zejména u léčiv, protože biologická aktivita závisí na prostorové konformaci molekuly a vzájemné pozici částí molekuly s určitými vlastnostmi. LBVS umožňuje použít metody strojového učení [3] a je výpočetně efektivnější než SBVS. V neposlední řadě je také jedinou možností v případě, kdy není známá struktura cílové makromolekuly.

Jak tedy zadefinovat molekulovou podobnost? Možností je velmi mnoho. Z přístupů používajících 3D reprezentaci sloučeniny stojí za zmínku metody založené na společném objemu či povrchu molekul. Tyto metody spočívají ve vypočítání polohy jednotlivých atomů v prostoru a následném hledání takového posunutí a natočení molekul, že překrývající se části mají maximální objem. Velká objemová shoda by pak měla značit, že mají molekuly velmi podobné prostorové uspořádání[4]. Dalším přístupem je hledání molekul splňujících soubor specifických chemických vlastností potřebných pro danou biologickou aktivitu. Také soubory vlastností se nazývají **farmakofory**.[82]

Mnohem častější jsou metody pracující se zjednodušenou reprezentací molekul odvozenou z vlastností dané molekuly. Takové řetězce se v odborné literatuře označují jako **molekulární otisky prstů** *(fingerprints, FP)*. Toto označení je poměrně přiléhavé, jelikož FP je soubor vlastností, které molekula má či nao-

[3] Nezbytnou podmínkou pro použití strojového učení je znalost většího množství aktivních molekul.

[4] viz aplikaci tohoto principu v OpenEyeToolkitu[5]

pak nemá a tvoří tak její otisk. Jednotlivé typy FP se liší v tom, co znamenají jednotlivé položky v řetězci. Běžné je použití sady charakteristických kusů molekul o typické velikosti v jednotkách atomů. Mezi nejčastěji používané FP topologické a cyklické FP a také FP založené na podstrukturách molekuly a jejích farmakoforech.[18] Podrobněji se FP metodami budeme zabývat v kapitole 1.

Cíl této práce

V této práci jsme si dali za cíl vyzkoušet odlišný způsob reprezentace molekul v LBVS. Grafová reprezentace je zřejmým mezikrokem mezi počítáním přesného tvaru molekuly a její reprezentací pomocí řetězce popisujícího jednotlivé fragmenty. Na rozdíl od přímého 2D zobrazení molekuly do plochy udržuje grafová reprezentace pouze důležité informace o typech atomů a jejich vazbách. Ostatně i díky tomu je grafová reprezentace molekul ve světě chemoinformatiky poměrně rozšířená. Většina molekulových formátů je založena na nějakém způsobu reprezentace molekulového grafu. Například u formátu MOL se jedná o seznam hran s přidanými řádky obsahujícími informace o atomech[7], u lidsky čitelného řetězcového formátu SMILES jde o zápis vzniklý průchodem molekulového grafu do hloubky[81], viz sekce 4.2.

Přestože se v jiných oblastech chemoinformatiky grafová reprezentace používá běžně, o použití ve virtuálním screeningu mnoho publikováno nebylo. Kvůli složitosti grafových algoritmů jsou metody užívající grafovou reprezentaci molekul pomalejší, než metody používající řetězcovou reprezentaci. Chtěli bychom však zjistit, zda jsou tyto metody konkurenceschopné v kvalitě výsledného výstupu a zda je výpočetní složitost jediným důvodem, proč se tyto přístupy v praxi ve virtuálním screeningu nepoužívají.

V této práci vyzkoušíme několik různých algoritmů zjišťování podobnosti grafů reprezentujících molekuly a použijeme je na testovacích datasetech. Nejdříve uvedeme stávající FP metody a následně ve druhé a třetí kapitole uvedeme algoritmy použité pro výpočet největšího společného podgrafu a grafové editační vzdálenosti. Poté v kapitole o implementaci popíšeme použité datové formáty. Následně v páté kapitole přistoupíme k rozporu provedených měření. Začneme obecným popisem průběhu virtuálního screeningu, popisem datasetů, převodem výsledků na podobnost a metodami vyhodnocení. Dále probereme specifika a parametry použitého softwaru a nakonec provedeme rozbor výsledků a srovnání námi zkoumaných metod se stávajícími metodami.

1. Stávající metody

V této sekci krátce popíšeme metody molekulárních otisků prstů, které se v současnosti v LBVS používají nejčastěji. Molekulový otisk specifikuje sadu molekulových fragmentů, které se v molekule nachází, či naopak nenachází.

Molekulární otisky prstů můžeme podle jejich charakteru rozdělit do následujících skupin[56]:

- Topologické (také zvané *path-based*) jsou definované pomocí cest vzniklých při průchodu molekulovým grafem. Molekulovými fragmenty jsou tedy různě dlouhé úseky některých takových cest. Příkladem tohoto druhu fingerprintů jsou AP a TT.

- FP založené na podstrukturách reprezentují molekulu pomocí různých podstruktur, které mají typicky menší velikost. Jako příklad této kategorie FP uvádíme MACCS.

- Cirkulární (nebo také Morganovy) FP patří mezi nejnověji vyvinuté[71] a jsou založené na reprezentaci okolí jednotlivých atomů v molekule do určitého poloměru. Mezi cirkulární FP patří různé varianty ECFP.

V této kapitole krátce představíme některé z metod molekulárních otisků prstů. V kapitole 5.6 je pak srovnáme s metodami, které jsme zkoumali.

1.1 AP

Atom Pairs (AP) definovali poprvé Carhart, Smith a Venkataraghavan[17]. Molekula je reprezentována pomocí dvojic atomů. U každé dvojice atomů máme uloženou informaci, jak dlouhá je cesta mezi nimi. Dvojice s délkou jsou poté převedené na čísla. Molekuly pak porovnáváme pomocí počtu shodných atomových párů.

1.2 TT

Topological Torsion (TT) reprezentuje molekulu pomocí molekulárních cest, které se skládají ze čtyř atomů a vazeb mezi nimi.[55] Autoři se při definování TT snažili o zachycení vlastností klíčových pro biologickou aktivitu. Přestože byl tento FP vymyšlen už v roce 1986, ve srovnávací studii se umístil mezi nejlepšími.[56, 57]

1.3 MACCS

Molecular ACCess System (MACCS)[41] je molekulární otisk vyvinutý v roce 1979 společností Molecular Design Limited, Inc. MACCS je definovaný pomocí 166 2D molekulových podstruktur.[27] Při vytváření tohoto FP procházíme molekulu a pro každou podstrukturu rozhodneme, zda se v molekule nachází či nikoli. MACCS byl původně vyvinutý pro účely vyhledávání podstruktur a jeho výsledky

v LBVS jsou spíše podprůměrné. Proto při porovnávání FP a jiných metod slouží jako základní smysluplná úroveň výkonu.[69].

1.4 ECFP

Na rozdíl od MACCS nebyl *Extended Connectivity Fingerprint (ECFP)* navržený pro hledání podstruktur. Stejně jako u TT bylo při návrhu hlavním cílem zaznamenat molekulární vlastnosti související s biologickou aktivitou. V případě ECFP je molekula reprezentována pomocí jednotlivých atomů a jejich paprskovitého okolí. Velikost okolí, tedy délka paprsků, se udává v počtu vazeb. ECFP s poloměrem x značíme jako ECFPx.

1.5 Převod na podobnost

Pro převod dvou molekulových FP na podobnost existuje více metod, viz článek [63]. Nejčastějším a podle některých studií[11] i nejvhodnějším převodem FP na podobnost[1] je Tanimoto koeficient[34]:

$$T_c(\frac{N_{ab}}{N_a + N_b - N_{ab}})$$

kde N_a, respektive N_b, značí počet fragmentů v molekule a, respektive b, a N_{ab} značí počet fragmentů, které mají molekuly a a b společné.

[1] v obecném případě

2. Největší společný podgraf

Jedním z možných přístupů při zjišťování grafové podobnosti je hledání **největšího společného podgrafu** *(Maximum Common Subgraph, MCS)* dvou grafů. Chemoinformatici používají ve stejném významu také pojem **největší společná podstruktura**. V následujícím textu proto budeme tyto dva pojmy volně zaměňovat.

Použít největší společný podgraf pro určení míry podobnosti dvou molekulových grafů může dávat dobrý smysl, jelikož kromě informace o míře podobnosti jsme schopní získat část nebo části molekul, které jsou zodpovědné za biologickou aktivitu.

Používání MCS je v chemoinformatice poměrně rozšířené. Aplikace nalezneme v dokování molekul,ve vyhledávání v chemických databázích, v predikování biologické aktivity, v modelování reakčních míst, a v interpretaci molekulárních spekter[61].

Dokonce najdeme i pár aplikací ve virtuálním screeningu. Duesbury, Holliday a Wilett[25] zkoumali použití MCS v LBVS. Kromě samostatného MCS zkoušeli použít i **hyperstruktury** *(Hyperstructure, HS)* vzniklé ze společné podstruktury obohacené o molekulové fragmenty původních molekul, viz obrázek 2.1.

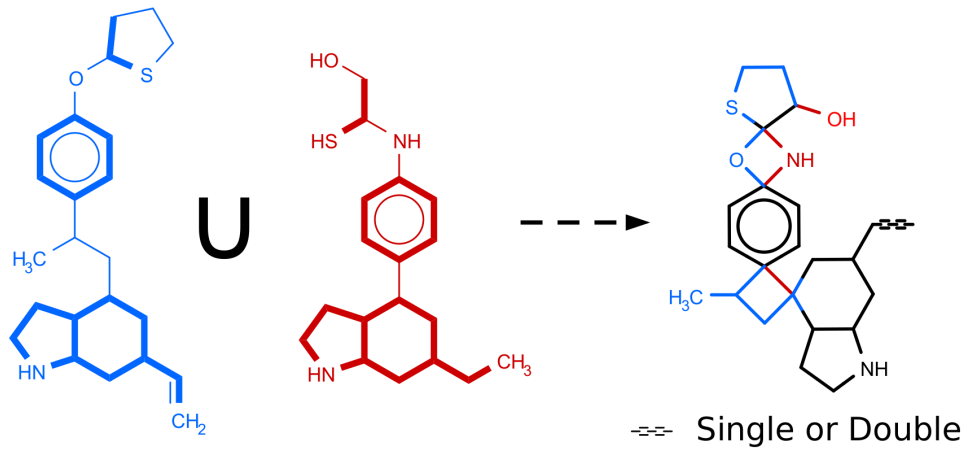
Autoři článku použili implementaci MCS ze třídy `MaxCommonSubstructure` v aplikaci JChem[1] ve verzi 6.1.0 (2014) od společnosti ChemAxon. Tato implementace je optimalizována na rychlé vracení přibližných výsledků. Výsledky srovnávali se standardním Morganovým FP (též známým pod zkratkou ECFP)[70] s poloměrem 3, který dle předchozích studií vykazoval velmi dobré výsledky ve virtuálním screeningu. U všech metod kromě HS bylo jako výsledná hodnota podobnosti bráno maximum z podobností vzhledem k jednotlivým aktivním látkám. Kromě spouštění základní varianty MCS, HS a FP zkoušeli autoři též kombinace jednotlivých přístupů. Při kombinování přístupů byly výsledné podobnosti sečteny.

Výsledkem studie bylo, že MCS dává celkově horší výsledky než zvolený FP. Aktivní látky nalezené pomocí MCS byly menší a obsahovaly méně heteroatomů, než ty nalezené pomocí FP. Celkový překryv molekul nalezených pomocí FP a MCS byl poměrně malý.

V článku však byl otestován pouze jeden způsob výpočtu MCS, který dával navíc pouze přibližné výsledky. Dále byly k otestování použity jen tři datasety. Rozhodli jsme se proto vyzkoušet jinou implementaci MCS na širším spektru datasetů, abychom ji mohli poté porovnat i s dalšími grafovými metodami. Na rozdíl od implementace použité ve zmíněném článku jsme použili implementaci, která vrací souvislou verzi MCS.

V této kapitole se nejdříve seznámíme s definicí MCS a existujícími algoritmy na jeho výpočet. Popíšeme testovanou implementaci a její parametry, které následně srovnáme v sekci s vyhodnocením výsledků.

[1] https://www.chemaxon.com/products/jchem-base/



Obrázek 2.1: Způsob tvoření hyperstruktury. Největší společná podstruktura je v původních dvou molekulách ztučněna. Ve výsledné hyperstruktuře je vyznačena černou barvou, barevně jsou pak vyznačené fragmenty původních molekul. Převzato z článku Duesburyho, Hollidaye a Willeta[25].

2.1 Definice pojmů

Dva grafy budou pro naše účely totožné, pokud budou isomorfní.

Definice 1 (**Isomorfismus grafů**). *Grafy $G(V_G, E_G)$ a $H(V_H, E_H)$ jsou izomorfní právě tehdy, když existuje isomorfismus $i : V_G \to V_H$, tedy že*

$$\forall v \in V_G \, \exists \, v' \in V_H : i(v) = v'$$

a

$$\forall v' \in V_H \, \exists \, v \in V_G : v' = i(v)$$

Zároveň musí pro hrany platit

$$\forall e = (x,y) \in E_G \, \exists \, e' = (i(x), i(y)) \in E_H$$

Dva grafy jsou tedy isomorfní právě tehdy, když existuje jednoznačné přeznačení vrcholů jednoho grafu na vrcholy druhého grafu při zachování hran.

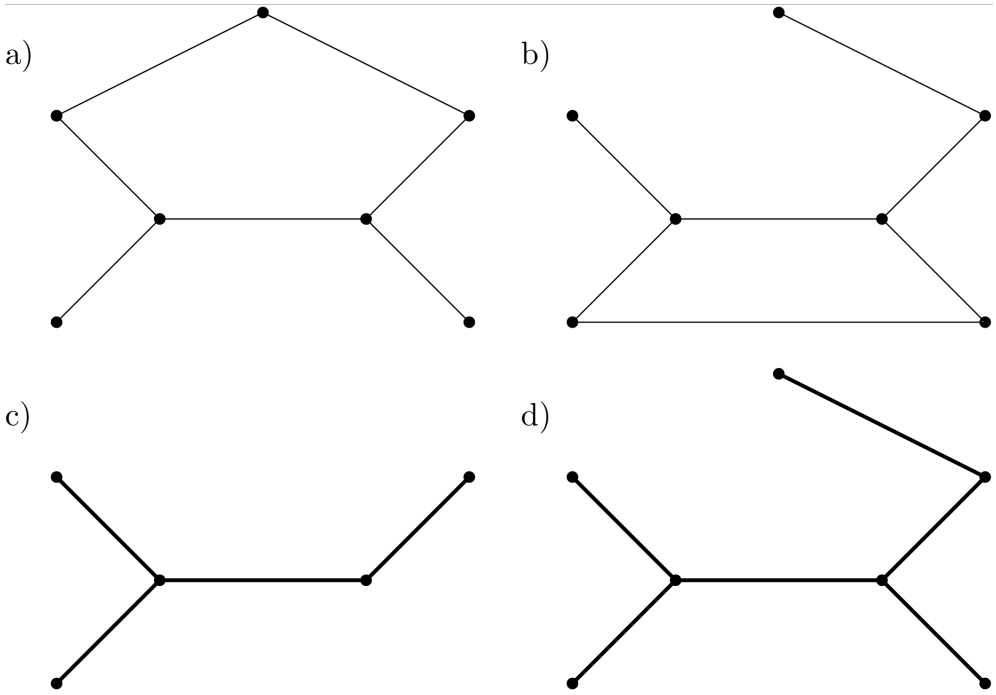
Pomocí grafového isomorfismu můžeme zadefinovat dvě varianty největšího společného podgrafu. Nejdříve uvedeme indukovanou variantu MCS.

Definice 2 (**Indukovaný podgraf**). *Graf $G_I(V_I, E_I)$ nazveme indukovaným podgrafem $G(V, E)$, pokud*

- $V_I \subseteq V$
- $\exists \, e = (x,y) \in E_I \iff x,y \in V_I \wedge \exists \, e = (x,y) \in E$

Definice 3 (**Největší společný indukovaný podgraf** (*Maximum Common Induced Subgraph, MCIS*)). *Mějme grafy G a H. Společným indukovaným podgrafem nazveme graf F, který je isomorfní nějakému indukovanému podgrafu G a nějakému indukovanému podgrafu H.*

Největším společným indukovaným podgrafem G a H pak nazveme společný indukovaný podgraf G a H, který je největší co do počtu vrcholů.



Obrázek 2.2: Rozdíl mezi MCIS a MCES. V částech a) a b) vidíme dva grafy. V částech c) a d) jsou po řade MCIS a MCES těchto dvou grafů.

Přestože je k této definici publikováno mnoho výsledků[62], v praxi úplně neodpovídá intuitivní představě o největší společné podstruktuře, jak je vidět na obrázku 2.2.

Hlavním důvodem nevhodnosti této definice je to, že právě vazby mezi atomy způsobují pozorovatelnou biologickou aktivitu.[51, 52] Proto uvedeme ještě druhou možnou definici, která více odpovídá intuici a představě o největší společné podstruktuře na molekulách.

Definice 4 (**Největší společný hranový podgraf** (*Maximum Common Edge Subgraph, MCES*))**.** *Společný podgraf grafů G a H je graf $F = (V_F, E_F)$, kde F je isomorfní nějakému podgrafu G a nějakému podgrafu H.*

Největší společný hranový podgraf grafů G a H je takový společný podgraf G a H, který má maximální počet hran.

Definici MCES pro množinu grafů obecné velikosti můžeme najdeme v článku [37]. V chemoinformatické literatuře je možné najít MCES také pod označením **maximální překrývající se množina** *(Maximum Overlapping Set, MOS)*[15]. Stojí za zmínku, že z výše zmíněných definic neplyne ani souvislost, ani jednoznačnost MCIS a MCES.

Při snaze využít největšího společného podgrafu na srovnávání molekul narazíme na několik podstatných detailů.

Abychom mohli použít tuto metodu pro porovnávání molekul, je potřebné nejdříve vyřešit způsob převedení molekuly na graf, respektive zadefinovat ekvivalenci na vrcholech. Při převodu je možné si pro každý vrchol uložit informace o daném atomu či vazbě, například typ prvku, typ farmakoforu, vaznost atp. Zavedeme si tedy graf s označenými vrcholy a ekvivalenci na takto označených vrcholech.

Definice 5 (**Označení vrcholů a jejich ekvivalence**). *Graf $G = (V, E)$ nazveme grafem s označenými vrcholy, pokud existují značkovací funkce $w_{iG} : V \to L_i$, kde L_i jsou množiny značek, $i \geq 1$.*

Mějme grafy s označenými vrcholy G a H se značkovacími funkcemi $w_{iG} : V_G \to L_i$, $w_{iH} : V_H \to L_i$. Pak $u \in V_G$ a $v \in V_H$ jsou ekvivalentní (značíme $u \approx v$) právě tehdy, když $w_{iG}(u) = w_{iH}(v) \, \forall i$.

Definice 6 (**Označení hran a jejich ekvivalence**). *Graf $G = (V_G, E_G)$ nazveme grafem s označenými hranami, pokud existují značkovací funkce $w_{eiG} : V_G \times V_G \to L_i$, kde L_i jsou množiny značek, $i \geq 1$.*

Mějme opět grafy G, H s označenými vrcholy a se značkovacími funkcemi pro vrcholy a hrany $w_{eiG} : V_G \times V_G \to L_i$, $w_{eiH} : V_H \times V_H \to L_i$. Pak $e = (u, v) \in E_G$ je ekvivalentní $f = (x, y) \in E_H$ právě tehdy, když $u \approx x \wedge v \approx y \wedge w_{eiG}(u, v) = w_{eiH}(x, y) \, \forall i$.

Dodejme ještě, že rovnost u značek (prvků L_i) může být nahrazena nějakou vhodně definovanou ekvivalencí, například náležením do stejné skupiny prvků. Při porovnávání shody atomů a vazeb pak můžeme požadovat přesnou shodu pro některé vybrané vlastnosti, které si uložíme pro každý atom a vazbu. Zřejmě platí, že čím striktnější pravidla na shodu si vybereme, tím menší budou nalezené podgrafy a tím větší bude citlivost této metody. V této kapitole vyzkoušíme několik možných nastavení citlivosti při hledání největšího společného podgrafu.

Další důležitou volbou je algoritmus na hledání největšího společného podgrafu a jeho implementace. Jelikož má tento problém exponenciální časovou složitost[48], existuje kromě přesných algoritmů také několik heuristik. V našem případě je přípustné vyzkoušet i přesné algoritmy s větší výpočetní složitostí díky tomu, že porovnávané molekuly tvoří malé a poměrně řídké grafy. Konkrétními algoritmy a heuristikami se budeme zabývat v následující sekci.

2.2 Algoritmy

Algoritmů na hledání MCS existuje mnoho. Bohužel je situace poměrně nepřehledná, protože existuje mnoho možných definic MCS. Nejde jen o to, zda algoritmus vrátí MCIS nebo MCES, ale například i o to, zda výsledný podgraf bude souvislý či nikoliv, zda algoritmus vrací přesný výsledek, či jen heuristiku a také zda-li algoritmus umí nalézt MCS pouze pro dvojice grafů nebo i pro větší množiny grafů. [2] V této práci se zaměříme na přesné algoritmy výpočtu MCES.

Praktický význam MCES a existující řešení pro hledání MCIS vzbudil zájem o řešení umožňující nalezení MCES pomocí algoritmů na výpočet MCIS. Whitney[83] dokázal, že hranový isomorfismus grafů G a H je ekvivalentní isomorfismu grafů $L(G)$ a $L(H)$, kde $L(G)$ značí *line graph* grafu G.[3]

Definice 7 (Hranový isomorfismus[35]). *Hranový isomorfismus mezi grafy $G = (V_G, E_G)$ a $H = (V_H, E_H)$ je bijekce $i_e : E_G \to E_H$ splňující $e_1, e_2 \in E_G$ jsou*

[2] Zobecnění algoritmu pro více jak dva grafy není triviální. Například neplatí tvrzení, že MCS pro každou dvojici grafů obsahuje MCS celé množiny.[37, viz strana 791 článku] Je tedy možné, že algoritmy umožňující větší množiny grafů budou komplexnější a jejich optimalizace bude náročnější.

[3] Za předpokladu, že nenastane záměna K_3 a $K_{1,3}$, v odborné literatuře zvaná $\triangle Y$ záměna. Tuto situaci je však velmi snadné ošetřit a ve 2D molekulových grafech téměř nikdy nenastává.

incidentní se společným vrcholem z $V_G \iff i_e(e_1), i_e(e_2) \in E_H$ *jsou incidentní se společným vrcholem z* V_H.

Definice 8 (Line graph)**.** *Mějme graf* $G = (V_G, E_G)$*. Pro line graph* $L(G) = (V_L, E_L)$ *platí, že vrchol z* V_L *jednoznačně odpovídá nějaké hraně z* E_G*. Dva vrcholy z* V_L *spolu sousedí právě tehdy, když odpovídající hrany v* E_G *mají společný vrchol.*

Výše uvedený výsledek je možné použít na snadný převod mezi MCIS a MCES. Stačí použít místo původního grafu G graf $L(G)$.

2.2.1 Hledání největší kliky v asociačním grafu

Pro účely vysvětlení následujícího algoritmu zavedeme pojmy **největší klika v grafu** a **association graph**, v některých publikacích označovaný též jako **modular product graph**, který pro účely tohoto textu budeme označovat jako **asociační graf**.

Definice 9 (Největší klika v grafu)**.** *Klikou v grafu G nazveme podgraf F grafu G, kde je každý vrchol spojený hranou se všemi zbylými vrcholy grafu. Grafy splňující tuto podmínku nazýváme úplné grafy a značíme* K_n*, kde n je počet vrcholů. Největší klikou v grafu G, neboli* $\omega(G)$*, nazveme takovou kliku, která má největší počet vrcholů.*

Definice 10 (Asociační graf *(Association Graph, Modular Product Graph)*)**.** *Asociačním grafem grafů* $G = (V_G, E_G)$ *a* $H = (V_H, E_H)$ *nazveme graf* $G \diamond H = (V_G \times V_H,\ E_{G \diamond H})$*, kde* (u_1, v_1) *a* (u_2, v_2) *spolu sousedí jestliže*

$$(u_1, u_2) \in E_G \wedge (v_1, v_2) \in E_H$$

nebo

$$(u_1, u_2) \notin E_G \wedge (v_1, v_2) \notin E_H$$

MCES mezi dvěma grafy odpovídá největší klice v asociačním grafu, viz strana 307 v článku [65]. Hledání největší kliky v grafu je také NP-úplný problém, ale tato úloha je mnohem známější, díky čemuž je k dispozici více algoritmů na jeho řešení.[59][58]

Algoritmus založený na této myšlence je použitý například v implementacích TopSim[26] a RASCAL[64]. Rekurzivního postupu pro hledání maximální kliky využili Bron a Kerbosh[14].

2.2.2 Generování podstruktur

Dalším možným přístupem pro získání MCES je generování všech podgrafů a následný výběr největšího. Armitageová a Lynch[10] byli pravděpodobně mezi prvními, kdo použil tento postup v chemoinformatice. Generování probíhá postupným zvětšováním podstruktur. Začínáme se souvislými podstrukturami obsahujícími dva atomy, tedy se všemi dvojicemi atomů spojenými vazbou. Následně jsou vyřazeny všechny podstruktury, které se neshodují v atomu či vazbě. Zbylé podstruktury jsou rozšířeny o další atom všemi možnými způsoby a postup se opakuje.

V tomto algoritmu hraje velkou roli reprezentace podstruktur. Je důležité, aby byla zvolena jednoznačná reprezentace. Tím je umožněné rychlé srovnávání molekul [23].

Základní verze algoritmu byla vylepšena Varkonym, Shiloachem a Smithem[80] a Takahashim a kol.[77]. Dalke a Hastingsová[21] pak implementovali tento algoritmus včetně vylepšení od Varkonyho a Takahashiho. Zároveň vymysleli a implementovali vlastní opatření zrychlující algoritmus. Tato implementace se stala pod názvem `fmcs` součástí chemického nástroje RDKit[6].

3. Grafová editační vzdálenost

3.1 Úvod

Editační vzdálenost na grafech je jedním z intuitivních přístupů ke zjišťování podobnosti grafů. Při počítání editační vzdálenosti se snažíme z jednoho grafu získat pomocí posloupnosti editací druhý graf. Editací rozumíme změnu označení a přidání nebo odebrání vrcholu či hrany. Editační vzdáleností pak rozumíme součet cen všech editací, tedy počet přidaných, odebraných či přejmenovaných hran a vrcholů přenásobený cenou dané operace. Sanfeliu a Fu[72] byli první, kdo grafovou editační vzdálenost formálně zadefinovali.

Ke spočítání této vzdálenosti je potřeba získat posloupnost editací. Ta může být definována pomocí přiřazovací funkce vrcholů jednoho grafu na vrcholy druhého grafu. Na rozdíl od grafového isomorfismu (viz definice 1) jsou definiční obor i obor hodnot rozšířeny o prázdný vrchol značený ϵ.

Definice 11 (Rozšířená přiřazovací funkce*(Error-Tolerant Graph Matching)*[66])**.** *Mějme grafy $G(V_G, E_G)$ a $H(V_H, E_H)$. Pak $f : V_G \cup \{\epsilon\} \to V_H\{\epsilon\}$ je rozšířená přiřazovací funkce, pokud je f bijekce pro všechny vrcholy V_1 a V_2, které nejsou obrazem ani vzorem ϵ. Formálně funkce f splňuje následující:*

- $f(x) \neq \epsilon \implies f(x) \neq f(y) \,\forall\, x, y \in V_G$
- $f^{-1}(u) \neq \epsilon \implies f^{-1}(u) \neq f^{-1}(v) \,\forall\, u, v \in V_H$

Řekneme, že jsme odebrali vrchol $x \in V_G$, pokud $f(x) = \epsilon$ a že jsme přidali vrchol $v \in V_H$, pokud $f(\epsilon) = v \iff f^{-1}(v) = \epsilon$. Zbylé vrcholy byly přeznačeny. Pro lepší čitelnost budeme $f(u) = v$ v následujícím textu značit jako $u \to v$.

Definice 12 (Editační operace na hranách[66])**.** *Mějme dvojice vrcholů $x, y \in V_G \cup \{\epsilon\}$ a $u, v \in H \cup \{\epsilon\}$, takové, že $f(x) = u$ a $f(y) = v$. Pokud*

- *$e_G = (x,y) \in E_G$ a $e_H = (u, v) \in E_H$, vznikla hrana e_H přeznačením hrany e_G.*
- *$e_G = (x,y) \in E_G$ a $e_H = (u, v) \notin E_H$ nebo $u = \epsilon$ nebo $v = \epsilon$, byla smazána hrana e_G.*
- *$e_G = (x,y) \notin E_G$ a $e_H = (u, v) \in E_H$ nebo $x = \epsilon$ nebo $y = \epsilon$, byla přidána hrana e_H.*

Pro účely posuzování podobnosti dvou grafů bereme hodnotu editační vzdálenosti, která odpovídá posloupnosti editací s nejnižší cenou. V tomto významu budeme používat pojem editační vzdálenost.

Volba ceny pro jednotlivé druhy editací hraje nezanedbatelnou roli při srovnávání posloupností grafových editací. Proto budeme požadovat, aby ceny editací splňovaly následující podmínky:

- nezápornost. Pokud by cena některých editací byla záporná, vycházely by jako optimální nekonečně dlouhé posloupnosti editací. Nulovost dále dovolíme pouze pro přeznačování vrcholů, ceny přidávání a odebírání vrcholů musí být kladné.

- symetrii, tj. stejnou cenu pro odebrání a přidání vrcholu nebo hrany. Díky této vlastnosti bude vycházet editační vzdálenost stejně pro transformaci v obou směrech.

- trojúhelníkovou nerovnost. Pro ceny editací dané funkcí c z editací do reálných čísel musí platit $c(u \to v) \leq c(u \to x) + c(x \to v)$, kde $u \to v$ znamená přeznačení vrcholu u na vrchol v. Díky tomu je zaručeno, že pro dvě posloupnosti editací p a q vedoucí ke stejnému výsledku platí, že pokud je p kratší než q, tak i $c(p) \leq c(q)$.

- identitu nerozpoznatelných, tj. přeznačení vrcholů se stejným označením má nulovou cenu.

Pokud jsou splněné výše uvedené podmínky, je editační vzdálenost metrikou.[74] Kromě toho podmínky omezují množinu editačních posloupností, kterou je nutné prohledat, abychom našli tu s nejmenší cenou. Přesto je tento problém NP-úplný.[84]

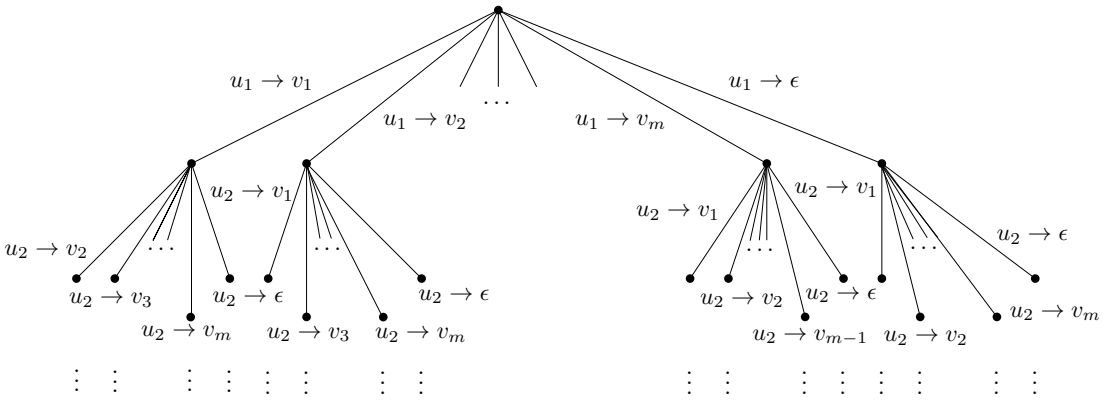
Existuje více přístupů k počítání **grafové editační vzdálenosti** *(Graph Edit Distance, GED)*, souhrn je možné nalézt v článku [33]. Některé metody řeší editační vzdálenost převodem na ekvivalentní, ale jinak formulovaný, problém a následným vyřešením tohoto problému již známými metodami, například pomocí lineární optimalizace[45] či neuronových sítí[53]. Existují i algoritmy používající MCS.[16, 73] Můžeme nalézt také algoritmy, které počítají editační vzdálenost na specifických třídách grafů, například na rovinných grafech[54], na grafech majících uspořádání na vrcholech[42], na grafech s unikátním označením vrcholů[22] a na stromech[78, 85]. Takové algoritmy využívají vlastnosí dané třídy grafů a díky tomu mají nižší složitost, než algoritmy pro obecné grafy.

V literatuře se nám nepovedlo najít významnější zmínky o použití grafové editační vzdálenosti v LBVS. Výjimkou je jen práce R. Sayla a kol.[73], ve které počítají GED pomocí MCES. Jejich metoda výpočtu MCES je rozdílná v tom, že obsahuje předgenerování, setřízení a uložení všech podstruktur všech molekul z databáze. Největší společná podstruktura je pak nalezena pomocí prohledávání setřízených posloupností. Tento přístup je zajímavý tím, že s rostoucí velikostí datasetu se snižuje doba výpočtu na jednu molekulu. V současnosti navíc již není problémem nedostatečná kapacita úložiště pro uložení předpočítaných podstruktur. V konferenčním příspěvku[73] autoři uvádí, že jejich metoda měla lepší výsledky než ECFP[70] s poloměrem 4. V této kapitole vyzkoušíme odlišný přístup k výpočtu GED. Použitá implementace umožňuje měnit poměr ceny editací vrcholů a hran. Na rozdíl od Sayla a kol. budeme kromě typu prvku uvažovat i jiná označení vrcholů.

3.2 Editační vzdálenost na obecných grafech

Pro užití v LBVS jsme se rozhodli pro klasický algoritmus procházení prostoru přípustných editačních posloupností. Důvodem byla snaha omezit závislost algoritmu na nastaveních nesouvisejících s daným problémem a vzájemnou závislost jednotlivých grafových přístupů.

Zvolili jsme implementaci *Graph Matching Toolkit (GMT)* od Kaspara Riesena[68], protože zahrnuje několik různých algoritmů na výpočet GED. Pro-



Obrázek 3.1: Editační strom. Inspirováno obrázkem 2.2 v práci Kaspara Riesena[66].

stor možných editací budeme uvažovat ve formě editačního stromu, viz obrázek 3.1. Každá větev stromu odpovídá posloupnosti editací vrcholů grafu $G = (\{u_1, u_2, ..., u_n\}, E_G)$ na vrcholy grafu $H = (\{v_1, v_2, ..., v_m\}, E_H)$ a ϵ. Aplikováním posloupnosti editací tedy nemusí být celý graf H, ale i některý z jeho podgrafů. V každé hladině stromu je rozhodnuto o editaci vrcholu u_i pro $1 \leq i \leq m$.

3.2.1 A*

Pro výpočet přesné grafové editační vzdálenosti používá GMT procházení stromu možných editací pomocí algoritmu A*[38]. Tento algoritmus prochází zadaný graf od počátečního vrcholu s a snaží se najít nejlevnější cestu do některého vrcholu z množiny zadaných cílových vrcholů T. V našem případě je s kořen editačního stromu a T jsou takové listy editačního stromu, jejichž posloupnosti odpovídají editaci G při které vznikne H.

Při hledání nejlevnější cesty hraje důležitou roli ohodnocovací funkce f. Hledání probíhá následovně:

1. Přidej s do otevřených vrcholů a ohodnoť ho.

2. Vyber otevřený vrchol v s nejnižším ohodnocením.

3. Pokud $v \in T$, označ ho jako uzavřený a skonči.

4. Jinak označ v jako zavřený. Vezmi všechny jeho syny, ohodnoť je a přidej do otevřených. Pokračuj bodem 2.

Ohodnocení $f(v)$ vrcholu v se skládá ze složek $g(v)$ a $h(v)$, kde $g(v)$ odpovídá ceně optimální cesty z s do v a $h(v)$ odpovídá ceně nejlevnější cesty z množiny cest mezi v a vrcholy $t \in T$. V průběhu algoritmu však neznáme tyto ceny přesně a proto se používají jejich aproximace. Aproximací $g(v)$ je minimální cena cesty z s do v, kterou algoritmus do té doby našel. Kromě ceny editací vrcholů se do této ceny započítávají i všechny editace hran způsobené dosud proběhlými editacemi vrcholů. Jako aproximaci $h(v)$ stačí použít libovolný dolní odhad. Důkaz toho, že poté algoritmus najde cestu s nejnižší cenou, je uvedený v článku[38].

Přesný výpočet GED je bohužel velmi časově náročný i pro relativně malé molekuly. Použitá implementace je v praxi schopná spočítat GED přesně pro grafy s maximálně dvanácti vrcholy (viz poznámka pod čarou na straně 145 v článku [68]). Kvůli tomu jsme nakonec vyzkoušeli v LBVS pouze heuristiky na výpočet GED.

3.2.2 Beam

Aproximace nazvaná Beam je založená na přesné metodě výpočtu pomocí A*. Na rozdíl od A* varianty je v algoritmu Beam zavedeno omezení na počet otevřených vrcholů. Čím vyšší je tento limit, tím se zvyšují přesnost výsledku a doba výpočtu.

Toto omezení je možné algoritmu nastavit pomocí konfiguračního souboru. V našem případě jsme použili omezení na 20 otevřených vrcholů.

3.2.3 Hungarian matching

Přibližný výpočet GED je možný také za pomoci algoritmů řešících optimalizační úlohu zvanou **assignment problem**[30], kterou můžeme převést na hledání maximálního párování s minimální cenou v bipartitním grafu s ohodnocenými hranami.

Definice 13. *Assignment problem (AP)*
Mějme množiny $A = \{a_1, a_2, ...a_n\}$ *a* $B = \{b_1, b_2, b_n\}$ *a matici* $\mathbf{C} = n \times n$, *kde* $c_{i,j} \in \mathbb{R}$ *udává cenu za přiřazení* a_i *k* b_j. *Úkol nalézt takovou permutaci* $p = p_1, ..., p_n$, *která minimalizuje* $\sum_{i=1}^{n} c_{i,p_i}$ *označujeme jako assignment problem.*

Assignment problem se dá převést na hledání maximálního párování s minimální cenou v úplném bipartitním grafu $G_{A,B}$ s ohodnocenými hranami, kde hodnota $c_{i,j}$ odpovídá ohodnocení hrany (a_i, b_j).

V našem případě rozšíříme matici $\mathbf{C}$ na rozměry $(n+m) \times (n+m)$, kde n a m jsou po řadě počty vrcholů grafů G a H. Přidané řádky a sloupce budou obsahovat ceny za odebrání či přidání vrcholu. Matice $\mathbf{C}$ je definována následovně[68]:

$$\mathbf{C} = \left[\begin{array}{cccc|cccc} c_{1,1} & c_{1,2} & \cdots & c_{1,m} & c_{1,\epsilon} & \infty & \cdots & \infty \\ c_{2,1} & c_{2,2} & \cdots & c_{2,m} & \infty & c_{2,\epsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n,1} & c_{n,2} & \cdots & c_{n,m} & \infty & \cdots & \infty & c_{n,\epsilon} \\ \hline c_{\epsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\epsilon,2} & \ddots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & \vdots \\ \infty & \cdots & \infty & c_{\epsilon,m} & 0 & \cdots & \cdots & 0 \end{array}\right]$$

Důležité je přitom stanovit hodnoty $c_{i,j}$ pro $i \leq n$ a $j \leq m$. Do $c_{i,j}$ přispívá cena za změnu atributu vrcholu. Potřebujeme však, aby se v $c_{i,j}$ promítly i editace hran, protože jinak by v algoritmu hrany vůbec nevystupovaly. Z toho důvodu se k $c_{i,j}$ přičte odhad na minimální cenu hranových editací způsobených přeznačením i na j. Tento odhad získáme opětovným spuštěním algoritmu, tentokrát na množiny hran e_i a e_j incidentní s i, respektive j.[67]

U hran bez atributů je cena přeznačení c_p pro všechny stejná. V takovém případě není nutné spouštět algoritmus znovu. Výsledek se dá spočítat jako $c_p \cdot \min(|e_i|,|e_j|) + ||e_i| - |e_j|| \cdot c_{ins}$, kde e_{ins} je cena za přidání, respektive odebrání vrcholu.

Pomocí této lokální aproximace na změny hran pak již pomocí některého z algoritmů řešícího AP najdeme aproximaci nejlepšího přiřazení vrcholů, dopočítáme editace hran a vrátíme spočítanou hodnotu grafové editační vzdálenosti. Tato hodnota nemusí být minimální.

AP řešíme pomocí tzv. maďarského algoritmu *(Hungarian algorithm, Hungarian matching)*[49], který vymyslel H.W. Kuhn na základě práce maďarského matematika Egerváryho[28]. V průběhu algoritmu postupujeme následovně:

1. Nejdříve v každém řádku najdeme minimum a odečteme jej od všech čísel v daném řádku. Důvodem je to, že celková částka bude vždy zvýšena o toto minimum, bez ohledu na přiřazení vrcholů. V každém řádku je nyní alespoň jedna nula.

2. Pokud existuje sloupec, který neobsahuje nulu, provedeme stejnou operaci pro daný sloupec. Opakujeme dokud není v každém sloupci alespoň jedna nula.

3. (a) Postupně procházíme řádky. Na začátku jsou všechny nuly použitelné. Pokud je v i-tém řádku použitelná nula na indexu j, provedeme přiřazení $i \to j$. Všechny zbylé použitelné nuly na řádku označíme jako nepoužitelné.

   (b) Dále si označíme všechny řádky, u kterých jsme neprovedli přiřazení. Také označíme všechny sloupce, které obsahují nulu v některém z nově označených řádků.

   (c) Přeškrtneme sloupce a řádky tak, abychom docílili přeškrtnutí všech nul pomocí co nejmenšího počtu čar. (Jedním ze způsobů je přeškrtnout označené sloupce a neoznačené řádky.)

4. Označme X množinu nepřeškrtnutých prvků a m_X minimum z této množiny. Odečteme m_X od všech prvků X a také od všech prvků přeškrtnutých v řádku i sloupci. Opakujeme kroky 3 a 4 dokud se nám nepovede provést přiřazení ve všech řádcích.

3.2.4 Volgenant-Jonker

Dalším algoritmem řešícím AP implementovaným v GMT je algoritmus Volgenanta a Jonkera[44]. **Algoritmus Volgenanta a Jonkera** *(VJ)* probíhá podobně jako maďarský:

1. Nejdříve v každém sloupci matice cen odečteme minimální hodnotu od všech prvků sloupce. Zprava doleva procházíme sloupce a pokud existuje ještě nepřiřazený řádek obsahující nulu v daném sloupci, tak je navzájem přiřadíme.

2. Dále chceme provést odečtení kladné hodnoty od každého řádku. V řádcích obsahujících nuly by tento postup však vedl ke vzniku záporných čísel. Mějme řádek i obsahující nulu na pozici j. K j-tému sloupci přičteme

nejmenší kladné číslo z řádku i. Tím jsme upravili řádek tak, aby šlo provést odečtení nejmenší kladné hodnoty. Upravíme tak všechny řádky a provedeme odečtení.

3. V této fázi hledáme střídavé cesty z nepřiřazených řádků k nepřiřazeným sloupcům. Střídavou cestou v tomto případě rozumíme posloupnost, kde se střídají řádky a sloupce.

   Cestu hledáme následovně: Zpracováváme nepřiřazený řádek i s nejmenší hodnotou min_1 a nejmenší hodnotou min_2 takovou, že $min_2 > min_1$. Vezmeme sloupce j_1 a j_2 obsahující po řadě hodnoty min_1 a min_2 v řádku i. Od řádku i odečteme min_2, čímž ve sloupci j_1[1] vznikne záporné číslo. Proto ke sloupci j_1 přičteme $min_2 - min_1$, abychom záporné číslo vynulovali. Bez ohledu na to, zda je sloupec j_1 přiřazený či nepřiřazený, přiřadíme jej k řádku i. Pokud byl sloupec j_1 přiřazený k řádku k, pokračujeme ve zpracovávání tohoto řádku, dokud se nepodaří narazit na nepřiřazený sloupec j_1.

4. V posledním kroku hledáme nejkratší střídavé cesty z nepřiřazených řádků do nepřiřazených sloupců. K tomuto účelu používá VJ upravenou verzi Dijkstrova algoritmu. Za cenu přechodu mezi řádkem i a sloupcem j považujeme $c_{i,j}$. Hledání probíhá dokud není nalezena nejkratší střídavá cesta pro každý nepřiřazený řádek.

Algoritmus jsme nastudovali v článku Jonese, Chawdharyho a Kinga[43].

[1] a samozřejmě zároveň také v řádku i

4. Implementace

4.1 Výběr chemoinformatické knihovny

Pro načítání formátů, zpracovávání molekul a vyhodnocování výsledků virtuálního screeningu je vhodné využít některou z existujících chemoinformatických knihoven. Předně jsme chtěli použít některou z open source knihoven, aby bylo možné zkontrolovat přesný způsob implementace použitých funkcí a případně opravit chyby. Dalším důvodem pro volbu open source byla přístupnost knihovny pro každého a tedy větší reprodukovatelnost.[36]

Dále jsme požadovali konverzi mezi běžně používanými formáty (SMILES, SDF), vyhodnocení výsledků virtuálního screeningu podle známých metrik a počítání známých fingerprintových metod pro možnost srovnání s grafovými algoritmy. Také jsme chtěli mít možnost vypsat pro daný atom seznam atomů, se kterými je spojený vazbou, abychom mohli provádět konverze do potřebných grafových formátů. Mezi další požadavky patřila dostatečná uživatelská i programátorská dokumentace.

Z dostupných chemoinformatických knihoven jsme vybrali RDKit[6], který splňoval všechny požadavky a navíc obsahoval i další funkce, které jsme se rozhodli využít.

Hlavním takovým přínosem byla třída `fmcs`, díky které jsme mohli otestovat rozumný algoritmus na největší společný podgraf bez nutnosti kombinovat několik různých jazyků a provádět export do dalšího formátu. Výhodou se také ukázala schopnost identifikovat farmakofory, spočítat 2D souřadnice atomů a vyexportovat obrázek molekuly. Ocenili jsme API pro Python.

V práci jsme také použili implementaci algoritmů na grafovou editační vzdálenost. Podrobnější informace k této implementaci uvádíme v sekci 5.5.

4.2 Formáty dat

Jak jsme již naznačili v úvodu, formátů pro reprezentaci molekul je mnoho. Většina formátů má své slabé a silné stránky v závislosti na zamýšlené aplikaci. Použití vhodné reprezentace molekul považujeme za místo, kde je potenciál docílit výraznějšího zrychlení vymyšlením reprezentací maximálně odpovídajícím požadavkům konkrétních grafových algoritmů. Chceme však podotknout, že níže popsané formáty jsou používány k přenosu informací o molekulách mezi jednotlivými programy a databázemi. Takové formáty jsou typicky nejdříve načteny a uloženy programem pomocí interní reprezentace, se kterou program dále pracuje. Dokud tedy neshledáme zásadní nedostatky, není důvod vymýšlet nové formáty tohoto typu.

Právě použitý interní formát dat může hrát ve virtuálním screeningu velkou roli, protože je potřeba přistupovat k velkému množství molekul ve velmi krátkém čase. Při porovnávání molekulových grafů hraje interní reprezentace molekuly ještě větší roli, než například při použití FP. U FP metod stačí většinou předpočítat FP a další manipulace s kompletní reprezentací celé molekuly již není nutná, jelikož FP samotný tvoří svého druhu interní reprezentaci molekuly. Grafové algoritmy však pracují s celými molekulami, což vytváří potřebu složitější interní

reprezentace sloučeniny.

Interní reprezentace jednotlivých programů však nejsou snadno dostupné a většinou je možné zjistit jejich specifika až přímo ze zdrojových kódů, jelikož většinou tvoří samostatné třídy objektů. Z těchto důvodů jejich popis vynecháváme.

Popíšeme zde stručně a zjednodušeně formáty k přenosu molekul použité v této práci a jejich specifika. Pokud nebude uvedeno jinak, použili jsme k parsování formátů knihovnu RDKit.

4.2.1 Molfile

Tento formát je jedním z nejrozšířenějších formátů v chemoinformatice. Autorem formátu je firma MDL Information Systems, Inc., nyní známá pod názvem BIOVIA[1]. Podrobnou specifikaci formátu publikovali Arthur Dalby a kol.[20]. Aktuální specifikaci je možné získat na stránkách společnosti[1].

MOL formát je tvořený hlavičkou, tabulkou s informacemi o atomech a seznamem vazeb a jejich vlastností. Jako oddělovače slouží odřádkování a mezery.

Hlavička obsahuje v prvním řádku název molekuly. V druhém řádku, kam patří poznámka, se typicky zmiňuje program, který molekulu do tohoto formátu exportoval, případně datum a uživatel nebo další obdobné informace. Další řádky slouží pro komentář.

První řádek **tabulky s informacemi o atomech** popisuje počet atomů, vazeb a verzi formátu. Kromě původní verze existuje verze V2000, která je aktuálně nejběžnější. Můžeme se setkat i s verzí V3000, kterou postupně začínají podporovat různé softwarové nástroje pro chemoinformatiku. Na každém dalším řádku najedeme informaci o jednom atomu. První tři sloupce popisují souřadnice atomu v prostoru.[2] Čtvrtý sloupec pak obsahuje typ prvku. Číslo řádku, na kterém je zapsána informace o atomu, budeme dále označovat jako číslo atomu.

Seznam vazeb v prvním sloupci obsahuje číslo vazby, v dalších dvou sloupcích jsou čísla atomů, mezi kterými se daná vazba nachází. Ve zbylých sloupcích najdeme informaci o typu vazby a další údaje.

Jako příklad nám poslouží obsah souboru .mol pro chloroform (viz obr. 4.1):

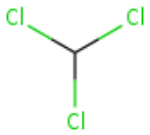
```
ZINC08214524
 OpenBabel03261619253D

  5  4  0  0  0  0  0  0  0  0999 V2000
 -0.0224    1.7987    0.0119 C   0  0  0  0  0  0  0  0  0  0  0  0
  0.0021   -0.0041    0.0020 Cl  0  0  0  0  0  0  0  0  0  0  0  0
  1.6691    2.4228   -0.0009 Cl  0  0  0  0  0  0  0  0  0  0  0  0
 -0.8944    2.3960   -1.4488 Cl  0  0  0  0  0  0  0  0  0  0  0  0
 -0.5326    2.1503    0.9087 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  1  4  1  0  0  0  0
  1  5  1  0  0  0  0
M  END
```

Přestože se jedná o jediný formát, je možné se s ním setkat ve dvou variantách. Základní varianta, kterou jsme popsali výše, má příponu .mol. Soubor s touto

[1] http://accelrys.com/products/collaborative-science/biovia-draw/ctfile-no-fee.html

[2] Jednotkou vzdálenosti je ångstrom.



Obrázek 4.1: Chloroform s kódovým označením ZINC08214524.

příponou obsahuje záznam o jedné molekule v popsané formě. Pro práci s větším množstvím molekul byla vytvořena ještě varianta označovaná **SDF** *(Structure Data File)* s příponou .sdf či .sd. SDF soubor obsahuje celou databázi molekul ve formátu MOL a umožňuje uložit k molekulám i další údaje. V této práci jsme formát SDF použili na reprezentaci molekul v datasetech.

Název údaje je na samostatném řádku vyznačený symboly „<“ a „>“. Hodnota údaje se nachází na dalším řádku. Jednotlivé molekuly jsou oddělené řetězcem „$$$$“ na samostatném řádku.

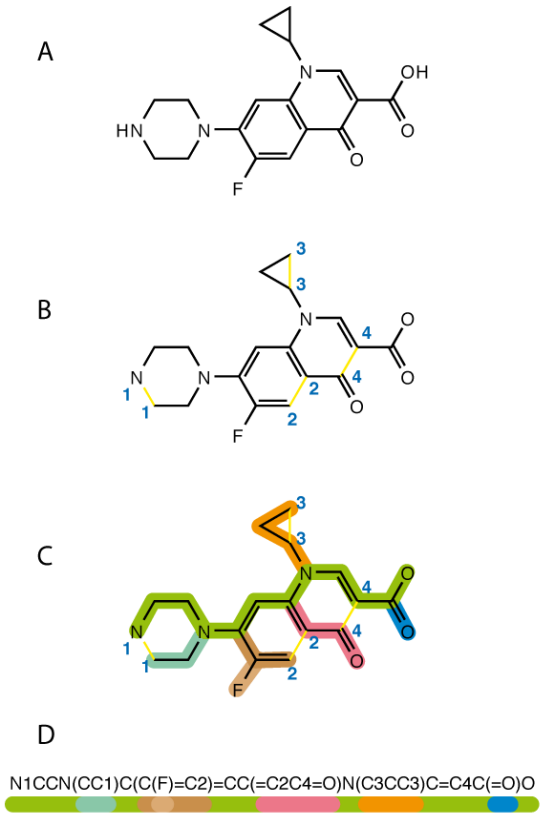
4.2.2 SMILES

Simplified Molecular-Input Line-Entry System (SMILES) vymyslel a popsal David Weininger v osmdesátých letech minulého století.[81]. O další vývoj formátu se pak starala zejména společnost Daylight Chemical Information Systems[3]. Před deseti lety vznikla otevřená varianta tohoto formátu s názvem **OpenSMILES**.

SMILES tvoří ASCII řetězec zakončený mezerou. Mezi hlavní cíle při tvorbě formátu bylo zaručit snadnou čitelnost pro lidi i počítače. Nakonec byl zvolený postup průchodu molekulárním grafem do hloubky. Od počátečního atomu postupně zapisujeme jednotlivé atomy a vazby. Když narazíme na cyklus, zrušíme jednu jeho vazbu tak, aby se z něj stal řetězec. K oběma atomům v místě rozpojení připojíme číslo, které postupně zvyšujeme. Větve zapisujeme do kulatých závorek. Když dojdeme k atomu, který už nemá žádné nenavštívené vazby, vracíme se stromem zpět k první neprojité větvi. Cestou zavíráme otevřené větve, které jsme už prošli kompletně. Příklad postupného vzniku SMILES reprezentace můžeme vidět na obrázku 4.2.

Zbývá doplnit způsob zapisování atomů a vazeb. Atomy jsou reprezentované pomocí písmene či písmen označujících daný prvek. První písmeno je vždy velké, druhé malé. Náboj, isomery a další doplňující informace je možné specifikovat obklopením symbolu prvku hranatými závorkami a doplněním informace bezprostředně za něj. Náboj se uvádí pomocí $+n$ či $-n$, kde n je malé písmeno. Vazby zapisujeme pomocí symbolů „-, =, #“ a „:“ po řadě pro jednoduché, dvojné, trojné a aromatické vazby.

Danou molekulu můžeme zapsat mnoha způsoby v závislosti na volbě počátečního atomu, pořadí, ve kterém navštěvujeme větve a způsobu rozdělení cyklů. Tato vlastnost je velmi nepříjemná zejména při snaze najít molekulu reprezentovanou pomocí SMILES v databázi. Kvůli tomuto a dalším problémům, které tato nejednoznačnost způsobila, vznikla snaha přidat pravidla zajišťující jednoznačnost. Výsledkem jsou reprezentace označované jako *canonical SMILES (cSMILES)*. Bohužel neexistuje žádná univerzální definice cSMILES. Mezi společnosti a softwarové nástroje s vlastním způsobem vytváření cSMILES patří Daylight Chemical



Obrázek 4.2: Postup tvorby SMILES.[32] V části A vidíme původní molekulu ciprofloxacinu. V části B jsou naznačena místa rozpojení řetězců. Část C názorně ukazuje průběh tvorby SMILES. Jednotlivé větve jsou barevně zvýrazněné. Publikováno pod licencí CreativeCommons[2].

Information Systems[3], OpenEye Scientific Software[5], RDKit[6], Indigo[4], Chemistry Development Kit[75] a mnoho dalších.

4.2.3 SMARTS

SMiles ARbitrary Target Specification (SMARTS) je formát založený na SMILES určený pro reprezentaci obecnějších struktur. Platí, že každá SMILES reprezentace je zároveň validní SMARTS. Navíc SMARTS přidává obecnější symbol „~“ pro libovolnou vazbu a možnost specifikovat libovolný prvek nebo podmnožinu prvků pro daný atom.

SMARTS se používá hlavně pro vyhledávání molekul a reprezentaci zobecněných molekulových grafů. Stejně jako SMILES byl tento formát vyvinutý Davidem Weiningerem a společností Daylight Chemical Information Systems[3]. Na stránkách této společnosti je také možné najít úplnou specifikaci.[3]

SMARTS je použito v této práci na reprezentaci MCS.

[3] `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`

4.2.4 GXL

GXL je typ XML používaný pro reprezentaci grafů. Pro reprezentaci molekulových grafů pro editační vzdálenost na obecných grafech jsme použili GXL s následujícím vzorem:

```
<gxl>
<graph id="NAZEV-SOUBORU.gxl">
<node id="_ID-VRCHOLU">
<attr name="NAZEV"><TYP>HODNOTA</TYP></attr>
    ⋮
</node>
⋮
<edge>
<edge from="_ID-POC-VRCHOLU" to="_ID-KONC-VRCHOLU">
<attr name="NAZEV"><TYP>HODNOTA</TYP></attr>
    ⋮
</edge>
⋮
</graph>
</gxl>
```

Hodnoty NAZEV-SOUBORU, NAZEV u atributu a HODNOTA u atributu si může tvůrce grafu zvolit na základě názvu molekuly, pojmenování jednotlivých vlastností a skutečných hodnot v grafu. Číselné identifikátory vrcholů musí být jednoznačné, identifikátory počátečního a koncového vrcholu pro hranu musí existovat. Za TYP je potřeba dosadit některý z povolených datových typů, například `string` pro řetězce, `float` pro desetinná čísla a `int` pro celá čísla.

Formát GXL popsali Hull, Winter a Schurr[39, 40]. Podrobnou specifikaci a tutoriál je možné najít na stránkách projektu[4]

4.2.5 CXL

GXL dokáže reprezentovat jeden graf, respektive molekulu, v jednom souboru. Pro reprezentaci množiny molekul používáme formát CXL, který je také odvozený od XML a vypadá následovně:

```
<?xml version="1.0"?>
<GraphCollection>
<graphs>
<print file="NAZEV-SOUBORU.gxl"/>
⋮
</graphs>
</GraphCollection>
```

CXL jsme použili jako doplnění GXL u implementace algoritmu na editační vzdálenost na obecných grafech.

4.2.6 JSON

Javascript Object Notation (JSON) je jednoduchý univerzální formát dat.

[4] `http://www.gupro.de/GXL/index.html`

Tento formát jsme používali pro ukládání a předávání konfigurace, ukládání mezivýsledků (např. výstupů z externích implementací) a exportování výsledků pro případné další zpracování.

JSON je lidsky editovatelný a čitelný. Základními stavebními prvky jsou objekty a pole. Následuje krátká ukázka syntaxe formátu JSON:

```
{
 "retezec" : "deset",
 "cislo" : 10,
 "pravdivostni hodnota" : true,
 "pole cisel" : [
    2,
    5
  ]
}
```

Objekty a pole je možné do sebe vzájemně vnořovat.

5. Testování grafových přístupů k výpočtu podobnosti

Před uvedením jednotlivých použitých implementací a získaných výsledků bychom rádi popsali způsob testování algoritmů a jejich vzájemného porovnávání. Detailů ovlivňujících výsledky testování je mnoho a pro účely reprodukovatelnosti a případný navazující výzkum je potřeba je popsat. Patří mezi ně průběh testování, charakteristiky testovacích datasetů, metody převodu výstupu algoritmů na podobnost, metriky využívané k vyhodnocení výsledků a použitý software.

5.1 Průběh měření

Nyní se dostáváme k samotnému průběhu LBVS. Na začátku máme zadanou databázi molekul k setřídění a seznam známých aktivních (a občas i neaktivních) látek. V našem případě je na vstupu (kromě parametrů pro podobnostní funkci) soubor ve formátu JSON[1] obsahující soubory s molekulami, seznam známých a testovacích molekul a další informace. Mezi dodatečné informace patří počty vybraných známých aktivních a neaktivních molekul, počty aktivních a neaktivních molekul v testovacím datasetu, číslo datasetu a metoda výběru. Seznamy známých aktivních i neaktivních molekul jsou tvořeny seznamem názvů molekul. Seznamy testovacích molekul tvoří kromě názvu i údaj o aktivitě, aby bylo možné následně vyhodnotit výsledek screeningu.

V první fázi načteme molekuly ze souboru .sdf a vytvoříme seznam aktivních molekul. Následně procházíme seznam testovacích molekul a počítáme podobnost vůči všem známým aktivním látkám. Jelikož pro seřazení molekul potřebujeme jediné číslo, vybereme z hodnot podobnosti tu největší. [2] Tento postup dává dobrý smysl, protože pro bioaktivitu stačí, aby byla molekula velmi podobná jedné aktivní látce a tedy se pravděpodobně vázala stejným mechanismem. Naproti tomu sledování podobnosti v průměru či mediánu nedává dobré výsledky, jelikož v souboru známých aktivních molekul může být více skupin molekul, z nichž každá se váže do aktivního místa jiným způsobem.

Poté vezmeme všechny testovací molekuly a seřadíme je podle získaného podobnostního skóre. Úspěšnost virtuálního screeningu se pak hodnotí pomocí různých metod, které popíšeme v sekci 5.4.

5.2 Datasety

Použili jsme datasety převzané z platformy na srovnávání datasetů, která obsahuje širokou škálu datasetů různých obtížností.[86] Vzhledem k náročnosti výpočtů jsme vybrali jen některé datasety. Zároveň jsme získali i výsledky z datasetů, které nejsou součástí základní varianty platformy. Kompletní seznam názvů všech datasetů tvoří vždy první sloupec tabulek s výsledky.

[1] `http://www.json.org/json-cz.html`

[2] Tento postup se v LBVS v angličtině označuje jako **max fusion**.

Platforma pro větší reprodukovatelnost a porovnatelnost výsledků obsahuje předgenerované výběry známých aktivních molekul. Výběry se liší v počtu zadaných aktivních a neaktivních molekul a také počtem aktivních látek přimíchaných v testovacím datasetu. Použili jsme výběry obsahující 30 zadaných aktivních látek. V testovacích datech bylo celkem 4900 molekul, z čehož 20 bylo aktivních.[3].

Použitá data a výběry aktivních molekul jsou součástí přílohy.

5.3 Převod na podobnost

5.3.1 MCS

Pro použití v LBVS jsme potřebovali převést MCES na podobnost molekul z intervalu $< 0, 1 >$. Základní převod by měl vrátit 1 pouze pro dva totožné grafy a blížit se nule pro grafy s malým MCES.

Výslednou podobnost grafů G a H jsme zavedli následovně:

$$\frac{|MCES|}{\max\{|V_G|,|V_H|\}}$$

pro $|MCES|$ značící počet vrcholů v MCES a $|V_G|$, $|V_H|$ počty atomů v porovnávaných grafech. U převodu jsme zvolili maximum z velikostí atomů v porovnávaných grafech. Při volbě minima bychom dostávali podobnost 1 pro graf a jeho libovolný podgraf.

5.3.2 GED

K převodu GED na podobnost používáme vztah

$$1 - \frac{GED}{\max_{GED}}$$

kde GED je výsledná editační vzdálenost a $\max_{GED}$ je horní odhad na GED. Tento odhad získáme pro grafy $G = (V_G, E_G)$ a $H = (V_H, E_H)$ pomocí

$$\alpha \cdot c_v \cdot (|V_G| + |V_H|) + (1 - \alpha) \cdot c_e \cdot (|E_G| + |E_H|)$$

kde α je zadaný poměr vrcholů a hran pro počítání celkové ceny a c_v, respektive c_e, cena za přidání či odebrání vrcholu, respektive hrany.

5.4 Používané metriky

Po provedení virtuálního screeningu je potřeba vyhodnotit výsledky. Hodnocení výsledků by mělo záviset na reálných potřebách v chemické a farmakologické praxi, aby se z virtuálního screeningu nestala jen soutěž teoretiků a programátorů bez možnosti aplikace. Základní požadavek na virtuální screening je umístit aktivní molekuly na co nejpřednější místa. Pro představu, běžná velikost databází

[3] V benchmarku jsou tyto výběry uložené ve složce `data/datasets/`OBTÍŽNOST`/selections/random_01_30_100_20_4900_M`, kde OBTÍŽNOST zastupuje název složky s datasety stejné průměrné obtížnosti

prohledávaných ve vývoji léčiv se pohybuje v řádu miliónů molekul. Při možnosti reálně otestovat několik tisíc sloučenin by měly metody virtuálního screeningu vykazovat velmi dobré výsledky již ve skupině prvních 0.1% molekul.[79]

V této sekci rozebereme metriky, které jsme použili při vyhodnocování výsledků grafových metod zjišťování podobnosti.

5.4.1 EF

Enrichment Factor *(EF)* je metrika, která udává, kolikrát více aktivních látek je v prvním zlomku z molekul setřízeného výběru oproti stejnému počtu náhodně vybraných molekul.[13] Tato metrika je dobrá v tom, že pro správnou volbu z pomocí ní dokážeme zjistit, jak dobré vlastnosti má naše metoda na začátku setřízeného úseku. Jak jsme již naznačili v úvodu této sekce, právě informace o začátku setřízeného výběru nás v praxi u virtuálního screeningu zajímá. Proto jsme zvolili jako jeden z parametrů pro vyhodnocení EF.

Nevýhodou EF však je, že jeho hodnota závisí na počtu aktivních látek v testovacím datasetu n a na velikosti testovacího datasetu N, jak ukážeme níže. Proto není vhodné na základě konkrétních hodnot EF srovnávat metody testované na datasetech, pro které se tato čísla liší. V námi prováděných měřeních měl náhodný výběr pro testování velikost 4900 molekul a obsahoval 20 aktivních sloučenin.

Na druhou stranu nám však EF pro různé hodnoty z dává poměrně dobrou představu o rozložení počtu aktivních látek na začátku setřízené skupiny molekul. Truchon a Bayly[79] uvádějí přesnou definici EF:

$$EF = \frac{\sum_{i=1}^{n} p_i}{z \cdot n} \qquad \text{pro} \qquad p_i = \begin{cases} 1 & \text{pro } r_i \geq zN \\ 0 & \text{pro } r_i < zN \end{cases}$$

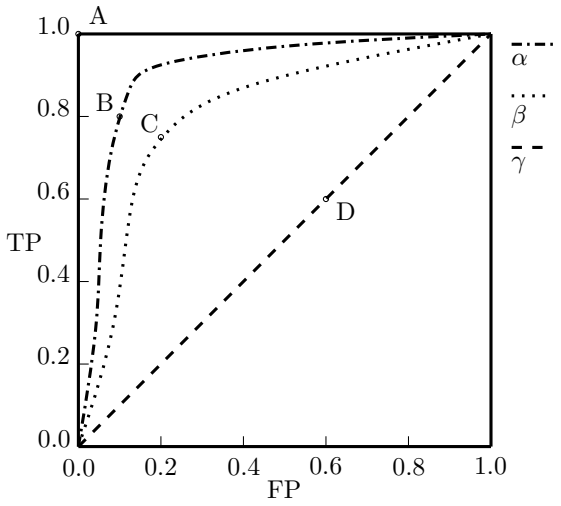
kde r_i je pořadí i-té aktivní molekuly. Suma v čitateli je počet aktivních molekul umístěných v první z-tině setřízené posloupnosti a číslo ve jmenovateli je z-tina aktivních molekul, z čehož jednoduše plyne slovní definice z předchozího odstavce.

Kromě výše zmíněné definice uvádějí Truchon a Bayly také minimální a maximální hodnoty EF. Minimální hodnota je 0 pro případ, kdy se do první z-tiny nedostanou žádné aktivní sloučeniny. Maximální hodnota $\max_{EF}$ závisí na parametru z následovně:

$$\max_{EF} = \begin{cases} \frac{1}{z} & \text{pro } z \geq \frac{n}{N} \\ \frac{N}{n} & \text{pro } z < \frac{n}{N} \end{cases}$$

V našem případě se maximální hodnota EF láme pro $z = \frac{20}{4900} \doteq 0.0041$. Pro nižší hodnoty z platí $\max_{EF} = \frac{4900}{20} = 245$, pro vyšší hodnoty bude $\max_{EF} = \frac{1}{z}$, tj. např. pro $z = 1\%$ je $\max_{EF} = 100$, ale pro $z = 50\%$ bude logicky $\max_{EF} = 2$, jelikož v náhodném výběru by ve střední hodnotě měla být polovina aktivních látek.

Jelikož EF zohledňuje výsledky pouze z první z-tiny, použili jsme ještě další běžně používanou metriku, která umožňuje vyčíslit kvalitu setřízení přes celou testovací sadu.



Obrázek 5.1: Ukázka ROC křivek.

5.4.2 AUC

Jedním ze způsobů zhodnocení dané metody je sestavení **ROC křivky** *(Receiver Operating Characteristic)*. Příklad ROC křivek s několika vyznačenými body můžeme vidět na obrázku 5.1. Každému bodu na křivce odpovídá nějaká hodnota prahu, pro který prohlásíme, že všechny molekuly s vyšší podobností jsou aktivní a všechny s nižší podobností jsou neaktivní. X-ová souřadnice bodu na křivce pak označuje podíl neaktivních molekul určených jako aktivní *(False Positive, FP)*. Y-ová souřadnice odpovídá podílu odhalených aktivních molekul *(True Positive, TP)*.[31]

Ideální hranicí by byl bod A ležící v levém horním rohu obrázku5.1. Při ní jsme správně rozpoznali všechny aktivní molekuly a žádnou neaktivní jsme neoznačili jako aktivní. Čím blíže tomuto bodu se křivka přibližuje, tím lepší je setřídění molekul. Tedy křivka α odpovídá lepšímu setřízení než křivky β a γ. Naopak body ležící na úhlopříčce nebo v její blízkosti označují takové setřídění molekul, které je téměř náhodné. Právě náhodnému rozložení molekul odpovídá křivka γ.

Běžně se ROC křivka používá jako vodítko k určení nejlepší hranice pro rozdělení položek do dvou kategorií ve chvíli, kdy se položky obou kategorií prolínají a závisí nějakým způsobem na zkoumané vlastnosti.[76]

ROC křivku však můžeme použít i ke srovnávání jednotlivých metod. Stačí si všimnout, že ROC křivka pro náhodu tvoří úhlopříčku čtverce a že křivky metod s lepšími výsledky než náhoda se nacházejí nad ní. Díky tomu je zajímavým měřítkem **plocha pod ROC křivkou** *(Area Under the ROC curve, AUC)*. Hodnota pro náhodný výběr je 0.5, čím lepší je metoda, tím větší má hodnotu. Hodnota ROC odpovídá (v našem případě) pravděpodobnosti, že bude náhodně vybraná aktivní sloučenina mít větší hodnotu podobnosti, než náhodně vybraná neaktivní látka.[12]

5.5 Použitý software

5.5.1 fmcs

Funkci `fmcs` z RDKitu jsme se rozhodli vyzkoušet v LBVS na testovacích datasetech. Autoři ji podrobili výkonnostnímu srovnání s implementací pojmenovanou SMSD od Rahmana a kol.[60], která v současnosti patří k nejčastěji používaným, a s implementací v Indigu[4]. [4] V odborné literatuře se nám bohužel nepovedlo nalézt uspokojivé srovnání různých existujících implementací. Nedostatek publikací srovnávajících stávající implementace na stejných sadách molekul zmiňují i Duesbury, Holliday a Willet[24].

`fmcs` je snadno dostupná a dobře zdokumentovaná implementace MCES. Zároveň je jedna z mála, u které existuje srovnání s některou další implementací, které vypovídá o rozumné časové náročnosti této implementace. Výstupem `fmcs` je souvislá varianta MCES.

Reprezentace molekul a MCS

`fmcs` používá interní reprezentaci molekuly z RDKitu. V této reprezentaci jsou mimo jiné dostupné informace o typu prvku, typu vazby a počtu atomů a vazeb. Pro daný atom či vazbu v molekule je dále možné zjistit, zda leží na cyklu. Pro úplný seznam dostupných metod nad objektem `Mol` z RDKitu viz `http://www.rdkit.org/docs/`

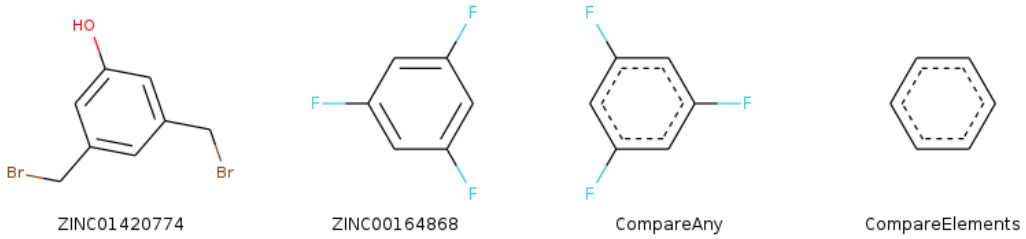
MCS je vraceno ve formě objektu `MCSResults`. Tento objekt umožňuje mimo jiné získat počet atomů a vazeb a reprezentaci ve formě SMARTS, který je dále možné konvertovat zpět na objekt třídy `Mol`, která je v RDKitu používaná jako interní reprezentace molekul.

Parametry

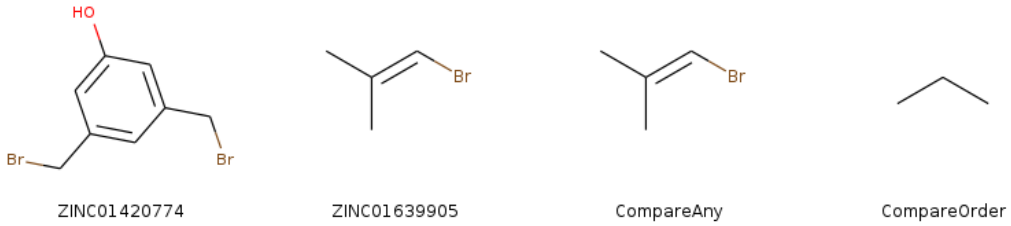
Jak bylo již nastíněno dříve, možných přístupů k převedení molekul na označené grafy je mnoho. Proto zavádí `fmcs` parametry umožňující nastavit citlivost srovnání atomů a vazeb. Pokud parametr `atomCompare` nastavíme na `CompareAny`, tak si při srovnávání budou odpovídat libovolné dva atomy. Citlivost můžeme zvýšit nastavením tohoto parametru na `CompareElements`, kdy si při srovnávání budou odpovídat atomy stejných prvků. Nejstriktnější volbou je `CompareIsotopes`, kdy si při srovnávání budou odpovídat pouze stejné isotopy. Srovnání jednotlivých nastavení parametru `atomCompare` viz obrázek 5.2

Nejvolnější nastavení parametru `bondCompare` je `CompareAny`, kdy se shodují libovolné dvě vazby. `CompareOrderExact` naproti tomu umožňuje vzájemné přiřazení pouze vazeb stejného řádu. Kompromisem mezi těmito variantami je volba `CompareOrder`, která dovoluje vzájemnou shodu jednoduchých a aromatických vazeb. Ve všech ostatních situacích musí být vazby stejného řádu. Srovnání jednotlivých natavení bondCompare viz obrázky 5.3 a 5.4

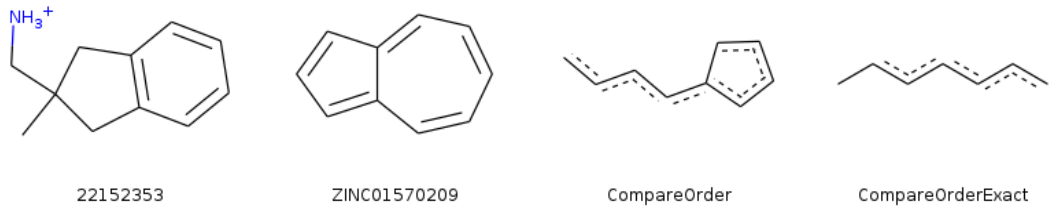
[4] `fmcs` dosahuje obdobných výsledků za 0,3- až 1,2-násobek času oproti implementaci v Indigu[21]



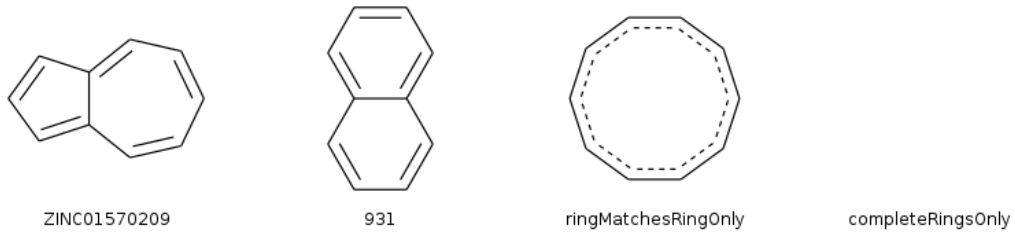
Obrázek 5.2: Rozdíl mezi volbou CompareAny a CompareElements u parametru atomCompare. Zleva vidíme původní molekuly, následuje MCS s volbou CompareAny a MCS s volbou CompareElements. U obou volání bylo nastaveno bondCompare na CompareAny a ostatním parametrům byly ponechány původní hodnoty.



Obrázek 5.3: Srovnání voleb `CompareAny` a `CompareOrder` parametru `bondCompare`. Parametr atomCompare byl nastaven na `CompareAny`, ostatním parametrům bylo ponecháno původní nastavení.



Obrázek 5.4: Porovnání voleb `CompareOrder` a `CompareOrderExact` parametru `bondCompare`. Díky záměnnosti jednoduchých a aromatických vazeb u `CompareOrder` se v odpovídajícím MCS objevuje i pěticyklus. Parametr `atomCompare` opět nastaven na `CompareAny`, `ringMatchesRingOnly` zapnuto.



Obrázek 5.5: V pravé části obrázku vidíme výstup algoritmu pro molekuly z levé části obrázku pro nastavení `ringMatchesRingOnly` a `completeRingsOnly`. Vidíme, že u prvního zmíněného nastavení jsme našli MCS podél obvodu cyklů. Druhé nastavení vrací prázdnou množinu, protože všechny vazby jsou součástí cyklů a neexistuje společný cyklus stejné velikosti.

Mezi další parametry patří

- `maximizeBonds` vedoucí k výběru MCS s největším možným počtem hran
- `matchValences` umožňující brát v úvahu vaznost atomů
- `ringMatchesRingOnly` díky kterému se atomy a vazby shodují jen tehdy, kdy oba atomy či vazby jsou, respektive nejsou, součástí nějakého cyklu. Tato volba zamezuje lineárnímu řetězci, aby byl přiřazen k cyklu a naopak.
- `completeRingsOnly` rozšiřuje předchozí variantu dalším omezením. Atomy v cyklu se shodují pouze když jsou součástí cyklu i ve výsledném MCS. Rozdíly mezi výstupem při nastavení `CompleteRingsOnly` a `ringMatchesRingOnly` viz obrázek 5.5
- `timeout` kterým je možné omezit čas výpočtu MCS. Po zvolené době v sekundách je vrácena největší společná podstruktura nalezená do uplynutí limitu. Zda byla nalezena přesná či pouze přibližná metoda je možné zjistit pomocí jednoho z atributů vráceného objektu.

Volání funkce `fmcs` jsme prováděli v jazyce Python pomocí RDKitového rozhraní.

5.5.2 GMT

GMT je napsaný v programovacím jazyce Java. Vstupem je konfigurační soubor .prop, který obsahuje nastavení ve formátu `parametr=volba` na samostatném řádku.

Parametry

Krátce okomentujeme jednotlivé parametry a jejich možná nastavení.

- `source`
  Cesta k souboru ve formátu CXL[5], který obsahuje množinu molekul ke srovnávání, v našem případě molekuly k otestování.

[5]Podrobnosti o formátu viz 4.2.5

- `target`
  Cesta k souboru ve formátu CXL, který obsahuje druhou množinu ke srovnávání, tedy skupinu známých aktivních látek. Program vypočítá GED pro každou dvojici molekul, kde jedna je ze `source` souboru a druhá z `target` souboru.

- `path`
  Cesta k souboru, kde jsou uložené GXL[6] soubory specifikované v souborech `source` a `target`.

- `result`
  Cesta, kam se má uložit soubor s výsledky.

- `matching`
  Tato volba udává, jaký typ algoritmu má být použitý pro výpočet GED. Možnosti jsou `AStar, Beam, Hungarian` a `VJ` (pro algoritmus Volgenant-Jonker).

- `s`
  Maximální počet najednou otevřených vrcholů pro algoritmus Beam. Při použití ostatních algoritmů se tento parametr nepoužívá.

- `adj`
  Při volbě `best` je při výpočtu ceny $c_{i,j}$ v matici $\mathbf{C}$ zavolán maďarský algoritmus na hranách incidentních s vrcholy i a j. Při volbě `worst` se cena určí pouze na základě ceny za přeznačení atributů vrcholů.

- `node` a `edge`
  Udávají cenu za vložení a odstranění vrcholu (parametr `node`) či hrany (parametr `edge`). Cena je zadávána číslem, necelá čísla je potřeba zadat s desetinnou tečkou. V souladu s podmínkami na ceny musí být zadané hodnoty kladné.

- `numOfNodeAttr` a `numOfEdgeAttr`
  Počet atributů vrcholů, respektive hran, v GXL souboru.

- `nodeAttr`i a `edgeAttr`i
  Název atributu i u vrcholu či hrany z GXL souboru.

- `nodeCostType`i, `edgeCostType`i
  Funkce, která má být použitá pro porovnávání atributu i. K dispozici jsou implementace funkcí `squared, absolute, discrete, sed`. `absolute` odpovídá klasické absolutní hodnotě, `squared` vrací druhou mocninu rozdílu atributů. Pro aplikaci `discrete` funkce je potřeba zadat parametry `nodeCostMu`i a `nodeCostNu`i pro každý atribut i, pro který se má tato funkce použít. `nodeCostMu`i udává cenu v případě, že jsou atributy shodné. Cena `nodeCOstNu`i se použije, pokud jsou prvky různé. `sed` označuje editační vzdálenost na textových řetězcích (*String Edit Distance, SED*) a je jedinou předpřipravenou funkcí, kterou je možné použít na atributy obsahující řetězec.

[6] Více o formátu viz sekce 4.2.4

Jelikož jsme chtěli použít i jiné funkce, implementovali jsme další funkce pro porovnávání atributů. Funkce `class` zjišťuje, zda prvky náleží do stejné třídy. Každý vrchol může náležet do více tříd. Všechny třídy musí být u daného atributu zadané pomocí řetězce, oddělené čárkami. Funkce prochází všechny třídy obou srovnávaných vrcholů a pokud mají společnou třídu, je jim přiřazena cena nastavená atributem `class`, jinak je použitá cena za odstranění či přidání vrcholu.

Další nově implementovanou funkcí je `given`. Tato funkce přiřadí porovnávaným prvkům cenu podle matice cen zadané pomocí atributů `atom`X jako seznam čísel oddělených čárkou a seznamu symbolů (X) oddělených čárkou `differentAtoms`.

- `nodeAttr`i`Importance`, `edgeAttr`i`Importance`
  Tento parametr udává relativní důležitost atributu. Zvolené číslo mezi 0 a 1 je koeficient, kterým se přenásobí cena za editaci atributu i, než se připočítá do celkové ceny za editaci vrcholů, respektive hran.

- `multiplyNodeCosts`, `multiplyEdgeCosts`
  Při volbě 1 se ceny za editaci jednotlivých atributů násobí, při volbě 0 se sčítají.

- `undirected`
  1 udává neorientovaný graf, 0 orientovaný. Pokud je graf zadaný jako orientovaný, bere se orientace hran z GXL souboru.

- `pNode`, `pEdge`
  Algoritmus vrací p-tou odmocninu z výsledné souhrnné ceny vrcholů, respektive hran.

- `alpha`
  Celková cena editací je spočítána jako $\alpha \cdot c_v + (1 - \alpha) \cdot c_e$, kde c_v a c_e jsou souhrnné ceny za editaci vrcholů a hran.

- `outputGraphs, outputEditpath, outputCostMatrix, outputMatching`
  Tyto parametry nastavují, co vše má program v průběhu vypisovat. 1 znamená vypisovat, 0 nevypisovat.

Atomům jsme se rozhodli přiřadit atributy odpovídající typu prvku, elektronegativitě a vaznosti. Hranám jsme přiřadili jediný atribut udávající násobnost vazby.

Výstup

Program původně vracel výstup ve strukturovaném textovém souboru. Pro naše účely jsme upravili třídu `ResultPrinter` v GMT, aby vracel výsledky ve formátu JSON[7] pro usnadnění dalšího strojového zpracování. To probíhá pomocí programu `evaluate-GED-screening.py`, který je možné, stejně jako ostatní programy a skripty zmíněné v této sekci, nalézt v příloze.

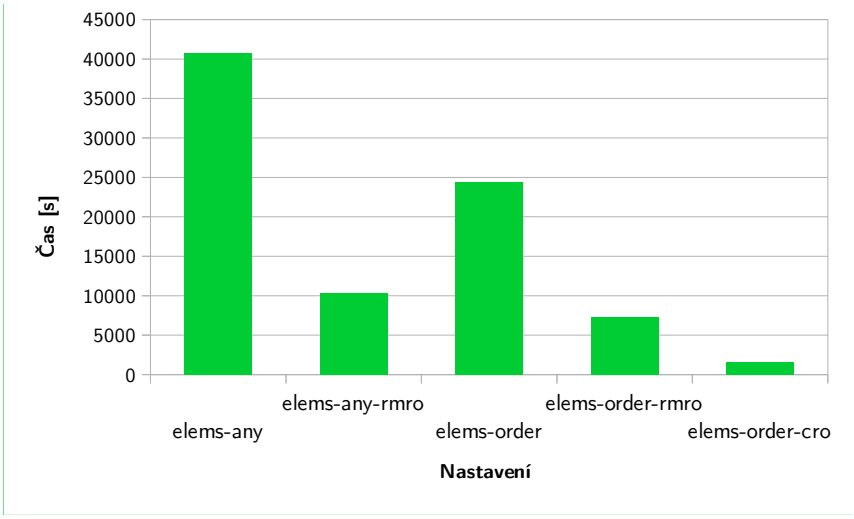
[7] Více o tomto formátu viz sekce 4.2.6

Volání a předávání informací

Abychom mohli spočítat GED, museli jsme provést konverzi molekul do formátu GXL. Tuto činnost provádí skript `convertToGXL.py`.

Vzhledem ke snadné dostupnosti většího počtu jader jsme chtěli, aby bylo možné jednotlivé splity v datasetu počítat paralelně. To obnášelo, aby se měnila cesta či název k souborům zadaným v `source` a `target`. Proto jsme zavedli vlastní konfigurační soubor ve formátu JSON pro zadávání parametrů. Vzhledem k počtu parametrů jsme se rozhodli do tohoto konfiguračního souboru umístit jen základní volby. Jednou z nich je cesta k plnému konfiguračnímu souboru, kde je možné doplnit ostatní volby. Parametry, které je možné vyplnit ve zkráceném konfiguračním souboru je nutné vyplnit v něm, volby ze souboru .prop se v tomto případě nepoužijí. Sestavení výsledného skriptu provádíme pomocí programu `awk`.

Všechny programy jsou spouštěné z funkce `run_ged_virtual_screening()`, která je součástí shellového skriptu `runVS.sh`.



Tabulka 5.1: Srovnání doby běhu jednotlivých nastavení `fmcs`.

5.6 Výsledky

V této sekci uvádíme výsledky vybraných grafových algoritmů ve virtuálním screeningu nad testovacími daty.

5.6.1 MCS

Kvůli výpočetní náročnosti a počtu splitů jsme se rozhodli vyzkoušet jen některé kombinace parametrů funkce `fmcs`. Snažili jsme se kombinace vybírat tak, aby rozumně zastoupily různě striktní požadavky na shodu. Pro `atomCompare` jsme se rozhodli vyzkoušet volby `CompareAny` a `CompareElements`. Volbu `CompareIsotopes` jsme nezkoušeli, protože se v datasetech nevyskytovaly různé druhy isotopů.

U parametru `bondCompare` jsme vyzkoušeli varianty `CompareAny`, `CompareOrder` a `CompareOrderExact`. Ze zbylých parametrů jsme pro kombinace parametrů s lepšími výsledky vyzkoušeli `ringMatchesRingOnly` a `comleteRingsOnly`. Přestože jsme to původně neplánovali, nakonec jsme nastavili `timeout` pro výpočty na 20 sekund. Toto rozhodnutí jsme provedli za účelem urychlení výpočtu, protože u některých molekul přesný výpočet trval příliš dlouho. V dokumentaci funkce `fmcs` je uvedeno, že pro některé molekuly může trvat výpočet v řádu minut.[8] Týkalo se to zejména méně striktních nastavení algoritmu a celkový počet molekul, kterých se tento limit týkal, byl minimální.

Obecně platí, že čím striktnější bylo nastavení, tím nižší čas byl k výpočtu potřeba, viz tabulka 5.1. Všechna nastavení běžela na počítači se dvěma osmijádrovými procesory Intel Xeon E5-2665 s frekvencí 2,40GHz pod operačním systémem Linux. Výpočet byl prováděn na jednom jádře, RAM byla omezena na 20 GB. Výše zmíněný výsledek dává dobrý smysl, protože s rostoucí přísností pravidel se snižuje počet molekul, které přicházejí v úvahu. Jako zástupce benevolentních pravidel pro shodu jsme chtěli vyzkoušet volbu `CompareAny` u `atomCompare` i

[8] `http://www.rdkit.org/RDKit_Docs.current.pdf`, strana 38 v dokumentaci pro verzi 2017.03.1

`bondCompare`. Bohužel však byl algoritmus při tomto nastavení tak pomalý, že se v rozumné době dopočítalo jen několik datasetů. Po prozkoumání výsledků z těchto datasetů jsme se rozhodli tuto kombinaci nastavení pro jeho neúspěšnost zavrhnout. Se stejnými problémy jsme se setkali i u kombinace `CompareAny` pro `atomCompare` v kombinaci s ostatními parametry.

V tabulce 5.2 vidíme souhrn AUC pro výběr z jednotlivých nastavení. Zobrazené hodnoty jsou průměrem z výpočtů na deseti předgenerovaných splitech, viz sekce 5.2. Kvůli velikosti a přehlednosti jsme nastavení parametrů vyjádřili zkráceně ve formě:

`atomCompare-bondCompare-dalšíParametry`

Jednotlivé volby jsme zkrátili následovně:

- `CompareElements` → elems
- `CompareAny` → any
- `CompareOrder` → order
- `RingMatchesRingOnly` → rmro
- `CompleteRingsOnly` → cro
- `matchValences` → valences

Parametr `maximizeBonds` byl vždy nastavený na `true`.

Tabulka 5.2 je rozdělena na 4 bloky podle průměrné obtížnosti datasetů u klasických metod. Právě podle obtížnosti jsme podbarvili jednotlivé buňky. Na obrázku 5.3 můžeme vidět přiřazení barev k výchylkám od očekávaného výsledku. Toto srovnání je pouze orientační. Přesné srovnání s konkrétními výsledky jednotlivých FP metod uvádíme v sekci 5.6.3.

Z tabulky 5.2 můžeme vidět, že lepší kombinace nastavení dávají na testovaných datasetech výsledky srovnatelné s průměrem. Můžeme si všimnout, že mezi nejlepší nastavení patří volba `ringMatchesRingOnly`. Při zachování ostatních parametrů tato volba znatelně zlepšuje AUC. Podobnou roli hraje i parametr `completeRingsOnly`. Zajímavou anomálií je zlepšení výsledků na datasetech s průměrnou obtížností $8{,}5 - 9{,}0$ oproti parametru `ringMatchesRingsOnly`. Naopak změna parametru `bondCompare` z `CompareAny` na `CompareOrder` výsledek spíše neovlivňuje.

Rozptyl mezi výsledky z jednotlivých testovacích výběrů se pohyboval okolo 0,05, výsledky tedy byly spíše vyrovnané. Celkově byl rozptyl nižší u lehkých datasetů (oddíl 9,8-1,0) a trochu vyšší u obtížnějších datasetů, což je očekávaný výstup. Celkově rozptyl nikdy nepřekročil hranici 0,1.

V tabulkách 5.4, 5.5 a 5.6 vidíme postupně EF pro horních 0,5 %, 1 % a 5 %. Podbarvení je tabulek je barevnou škálou od červené pro minimální hodnotu po zelenou pro maximální hodnoty. Minimum a maximum bereme přes výsledky ve všech třech tabulkách.

V tabulce 5.4 vidíme EF pro prvních 0,5 % molekul. $\max_{EF}$ je v tomto případě $\frac{1}{0{,}005} = 200$, jelikož $0{,}005 > \frac{20}{4900} \doteq 0{,}0041$. Právě na tomto výběru molekul s nejlepším skóre se může projevit vhodnost nebo naopak nevhodnost metod

	elems-any	elems-any-rmro	elems-order	elems-order-rmro	elems-order-cro
8.0 - 8.5					
5HT2B	0.764	0.817	0.787	0.809	0.820
5HT2C	0.763	0.851	0.770	0.843	0.804
ADA2A	0.729	0.825	0.754	0.826	0.822
CDK2	0.762	0.859	0.766	0.840	0.805
HDAC01	0.807	0.834	0.792	0.826	0.785
PXR_Agonist	0.870	0.909	0.885	0.911	0.903
8.5 – 9.0					
ACM1_Agonist	0.773	0.818	0.761	0.832	0.851
ADA2B_Antagonist	0.760	0.836	0.780	0.839	0.845
ADA2C_Antagonist	0.740	0.837	0.759	0.838	0.859
CHK1	0.887	0.947	0.851	0.934	0.882
9.0 – 9.5					
5HT1F_Agonist	0.842	0.920	0.837	0.908	0.881
DRD1_Antagonist	0.763	0.822	0.766	0.824	0.875
DRD2_Agonist	0.816	0.936	0.828	0.940	0.944
LSHR_Antagonist	0.878	0.947	0.882	0.949	0.919
OPRM_Agonist	0.885	0.909	0.892	0.911	0.946
9.8 – 1.0					
DHFR	0.970	0.985	0.978	0.988	0.986
MTR1A_Agonist	0.961	0.965	0.953	0.963	0.931
MTR1B_Agonist	0.946	0.943	0.944	0.942	0.907
P38	0.968	0.996	0.976	0.997	0.993
V2R_Antagonist	0.971	0.986	0.976	0.986	0.981

Tabulka 5.2: Srovnání AUC pro jednotlivé kombinace parametrů pro `fmcs`.

pro LBVS. V této tabulce je klíčové si všimnout rozdílu mezi dvěma nejlepšími metodami podle AUC, `elems-any-rmro` a `elems-order-cro`. Čistě podle hodnot AUC se totiž zdá, že `elems-any-rmro` je lepší nastavení než `elems-order-cro`, zejména na obtížnějších datasetech v horní části tabulky. Při srovnání EF na molekulách s nejvyšším skóre se však ukazuje, že na obtížnějších datasetech začíná vynikat právě nastavení `elems-order-cro`. S klesající obtížností datasetů se však rozdíly mezi metodami stírají. Naopak tvrzení, že volba `cro` má větší potenciál zlepšit výsledek než volba `rmro`, se ukazuje správné i pro molekuly s nejvyšším skóre.

V tabulce 5.5 nevidíme žádné dramatické změny v pozorovaných trendech. $\max_{EF}$ je tentokrát 100.

Na výsledcích v tabulce 5.6 je zajímavé, že v prvních 5 % už se rozdíly mezi jednotlivými nastaveními zdánlivě téměř stírají. Tento efekt je však způsobený tím, že v tomto případě je $\max_{EF} = \frac{1}{0{,}05} = 20$. Vidíme tedy, že `elems-any-rmro`

0.05 +
0 – 0.05
(-0.1) – 0
(-0.2) – (-0.1)
(-0.3) – (-0.2)

Tabulka 5.3: Barevné vyznačení odchylek od průměrné hodnoty.

a `elems-order-cro` se na jednoduchých datasetech P38 a V2R_Antagonist blíží maximální hodnotě, stejně jako tomu bylo i v předchozích tabulkách.

Z testování různých nastavení `fmcs` vychází jako nejlepší `elems-order-cro`. V sekci 5.6.3 jej srovnáme s výsledky FP metod.

5.6.2 GED

Z algoritmů implementovaných v GMT jsme vyzkoušeli Beam, maďarský algoritmus a algoritmus Volgenanta a Jonkera. Parametr α udávající poměr mezi cenou za editaci vrcholů a hran jsme zkoušeli s hodnotami 0,3, 0,5 a 0,8. Hodnota 0,3 dává větší váhu ceně hran, při volbě 0,8 se naopak na celkové ceně podílí větším dílem vrcholy. Hodnota 0,5 pak způsobuje, že se hrany i vrcholy do celkové ceny přispívají stejnou měrou.

Také jsme zkoumali vliv různých atributů vrcholů. Mezi atributy patřily typ prvku, elektronegativita a vaznost. Pro jednoduchost jsme atributy nekombinovali a při porovnávání vrcholů jsme brali v úvahu vždy jen jediný z nich. V tabulkách označujeme atributy následovně

- typ prvku → symbol
- elektronegativita → eneg
- vaznost → valence
- maďarský algoritmus → Hungarian

Pojďme nejdříve prozkoumat vliv volby parametru *alpha*. V tabulce 5.7 jsme pevně zvolili algoritmus na Beam a atribut vrcholů na elektronegativitu. Při této volbě vychází nejhůře, když se na celkové ceně více podílí hrany. Celkově jsou však výsledky podprůměrné, a to nejen při vyhodnocení pomocí AUC, neslavné jsou i hodnoty EF.

Z tabulky 5.8, ve které srovnáváme různé volby algoritmů právě při zafixované volbě $\alpha = 0{,}3$, však vidíme, že nízká volba α působí velmi příznivě v kombinaci s maďarským algoritmem. Ve srovnání s ostatními algoritmy vychází maďarský algoritmus nejlépe i při jiných volbách α, výsledky jsou však o trochu horší. Je možné, že algoritmus Beam dává horší výsledky v porovnání s maďarským algoritmem kvůli nízké volbě parametru `s`. Fankhauser, Riesen a Bunke tvrdí v článku [30], že použití VJ pro hledání GED místo maďarského algoritmu zrychluje výpočet a nemá velký vliv na výsledky. V našem případě se však ukazuje, že rozdíl mezi výsledky těchto algoritmů je znatelný. Při časovém srovnání v grafu 5.1 vidíme, že algoritmus Beam běží delší dobu, než maďarský a VJ, což svědčí o větší výpočetní náročnosti tohoto algoritmu. V souladu s výsledky Fankhausera a kol.[30] je algoritmus VJ rychlejší než maďarský. Výpočty byly prováděny pod operačním systémem Linux na procesoru Intel Core i7-6700 s frekvencí 3,40GHz s 15,5 GB RAM. Atributem byla elektronegativita a hodnota α byla 0,3.

V tabulce 5.6 si můžeme všimnout, že kombinace maďarského algoritmu a $\alpha = 0{,}3$ dává nadprůměrné výsledky i v kombinaci s jinými atributy.

5.6.3 Srovnání s ostatními metodami

V předchozích sekcích jsme srovnávali pouze jednotlivá nastavení metod mezi sebou. Nyní provedeme srovnání grafových přístupů k výpočtu podobnosti s klasickými FP metodami.

Pro všechny FP metody byl k výpočtu podobnosti použitý Tanimoto koeficient. ECFP měl velikost 1024 bitů. Jako zástupce MCS jsme vybrali nastavení `elems-order-cro`, GED zastupuje maďarský algoritmus s atributem elektronegativita a $\alpha = 0{,}3$.

V tabulce 5.10 najdeme souhrn AUC pro jednotlivé metody. V kombinaci s hodnotami EF s parametrem 0,5 % a 1 % můžeme vidět, že jsou grafové metody srovnatelné s klasickými FP metodami, přičemž MCS podává lepší výkon než GED.

Povedlo se nám tedy ukázat, že grafové reprezentace molekul jsou vhodné a použití grafových algoritmů pro počítání podobnosti mezi nimi dává výsledky srovnatelné se stávajícími metodami. Oproti otiskovým metodám přináší grafové přístupy nejen podobnost ve formě čísla, ale v případě MCS také zobecněnou představu o podobě aktivní molekuly.

	elems-any	elems-any-rmro	elems-order	elems-order-rmro	elems-order-cro
8.0-8.5	0.005	0.005	0.005	0.005	0.005
5HT2B	40.344	64.944	42.312	64.944	61.008
5HT2C	37.392	60.024	46.248	62.976	71.832
ADA2A	31.488	58.056	34.440	60.024	70.848
CDK2	32.470	56.084	35.422	55.100	63.955
HDAC01	31.488	42.312	29.520	44.280	48.216
PXR_Agonist	104.325	138.995	106.328	136.120	129.837
8.5 – 9.0					
ACM1_Agonist	27.551	55.103	28.535	55.103	67.895
ADA2B_Antagonist	35.424	64.944	41.328	64.944	82.656
ADA2C_Antagonist	40.344	57.072	45.264	59.040	78.720
CHK1	56.048	111.119	64.899	113.087	128.818
9.0 – 9.5					
5HT1F_Agonist	61.992	101.352	67.896	102.336	90.528
DRD1_Antagonist	52.152	63.960	42.312	68.880	84.624
DRD2_Agonist	57.072	98.400	62.976	98.400	99.384
LSHR_Antagonist	87.576	116.112	83.640	114.144	130.872
OPRM_Agonist	61.008	83.640	59.040	89.544	127.920
9.8 – 1.0					
DHFR	119.547	140.443	126.730	145.685	150.713
MTR1A_Agonist	102.336	137.760	108.240	137.760	136.776
MTR1B_Agonist	109.224	136.776	112.176	136.776	130.872
P38	145.930	161.166	154.410	164.267	173.655
V2R_Antagonist	130.872	182.040	133.824	182.040	175.152

Tabulka 5.4: Srovnání EF s parametrem 0,005 pro různé parametry `fmcs`.

	elems-any	elems-any-rmro	elems-order	elems-order-rmro	elems-order-cro
8.0-8.5	0.01	0.01	0.01	0.01	0.01
5HT2B	27.060	34.440	30.012	33.948	32.472
5HT2C	29.028	35.916	30.504	35.916	42.312
ADA2A	18.204	31.980	19.188	32.472	40.344
CDK2	19.679	32.962	25.090	33.453	36.897
HDAC01	21.648	27.552	19.680	28.536	31.980
PXR_Agonist	58.935	76.190	57.395	76.216	71.439
8.5 – 9.0					
ACM1_Agonist	18.696	30.503	20.664	33.455	41.819
ADA2B_Antagonist	26.076	37.884	25.584	38.376	46.740
ADA2C_Antagonist	27.060	32.964	25.584	32.964	44.280
CHK1	36.384	70.310	38.350	65.394	68.342
9.0 – 9.5					
5HT1F_Agonist	39.852	56.580	41.328	56.580	56.088
DRD1_Antagonist	30.012	37.392	27.552	38.376	51.660
DRD2_Agonist	34.440	53.628	38.868	55.596	53.628
LSHR_Antagonist	48.708	66.420	48.216	66.912	71.832
OPRM_Agonist	35.916	47.724	37.392	52.644	69.864
9.8 – 1.0					
DHFR	73.297	85.643	76.003	85.643	79.677
MTR1A_Agonist	69.864	77.244	69.864	76.260	72.816
MTR1B_Agonist	65.436	74.784	68.880	74.292	70.356
P38	82.403	89.685	84.447	91.791	93.951
V2R_Antagonist	74.292	93.972	74.784	93.972	89.544

Tabulka 5.5: Srovnání EF s parametrem 0,01 pro různé parametry `fmcs`.

	elems-any	elems-any-rmro	elems-order	elems-order-rmro	elems-order-cro
8.0-8.5	0.05	0.05	0.05	0.05	0.05
5HT2B	8.200	9.300	8.600	9.200	9.700
5HT2C	8.300	10.600	8.800	10.700	11.200
ADA2A	6.900	9.600	7.500	10.000	11.100
CDK2	6.499	10.699	7.099	10.499	10.699
HDAC01	7.500	11.000	6.700	10.400	10.600
PXR_Agonist	14.757	16.664	14.125	16.775	15.533
8.5 – 9.0					
ACM1_Agonist	8.200	10.100	8.100	10.600	12.300
ADA2B_Antagonist	8.800	10.400	9.100	10.800	12.700
ADA2C_Antagonist	8.100	9.400	8.400	9.400	11.400
CHK1	11.392	16.389	10.893	15.490	15.389
9.0 – 9.5					
5HT1F_Agonist	11.800	15.200	12.400	14.900	14.500
DRD1_Antagonist	9.600	10.000	9.900	10.400	14.000
DRD2_Agonist	10.900	13.700	11.400	13.600	14.800
LSHR_Antagonist	13.200	16.800	13.400	16.900	16.400
OPRM_Agonist	11.500	14.200	13.000	14.300	16.200
9.8 – 1.0					
DHFR	18.094	17.656	18.304	17.879	19.077
MTR1A_Agonist	17.700	17.200	17.500	17.000	16.400
MTR1B_Agonist	16.900	16.200	16.500	16.000	15.500

Tabulka 5.6: Srovnání EF s parametrem 0,05 pro různé parametry `fmcs`.

5HT2C	0.681	0.744	0.750
ADA2A	0.662	0.732	0.733
CDK2	0.564	0.579	0.541
HDAC01	0.476	0.522	0.583
PXR_Agonist	0.619	0.705	0.744
8.5-9.0			
ACM1_Agonist	0.764	0.820	0.755
ADA2B_Antagonist	0.704	0.772	0.760
ADA2C_Antagonist	0.665	0.723	0.721
CHK1	0.593	0.541	0.541
9.0-9.5			
5HT1F_Agonist	0.769	0.817	0.768
DRD1_Antagonist	0.707	0.783	0.796
DRD2_Agonist	0.742	0.780	0.784
LSHR_Antagonist	0.627	0.742	0.821
OPRM_Agonist	0.788	0.853	0.844
9.8-1.0			
DHFR	0.743	0.837	0.818
MTR1A_Agonist	0.908	0.898	0.853
MTR1B_Agonist	0.900	0.880	0.841
P38	0.766	0.814	0.831
V2R_Antagonist	0.843	0.881	0.894

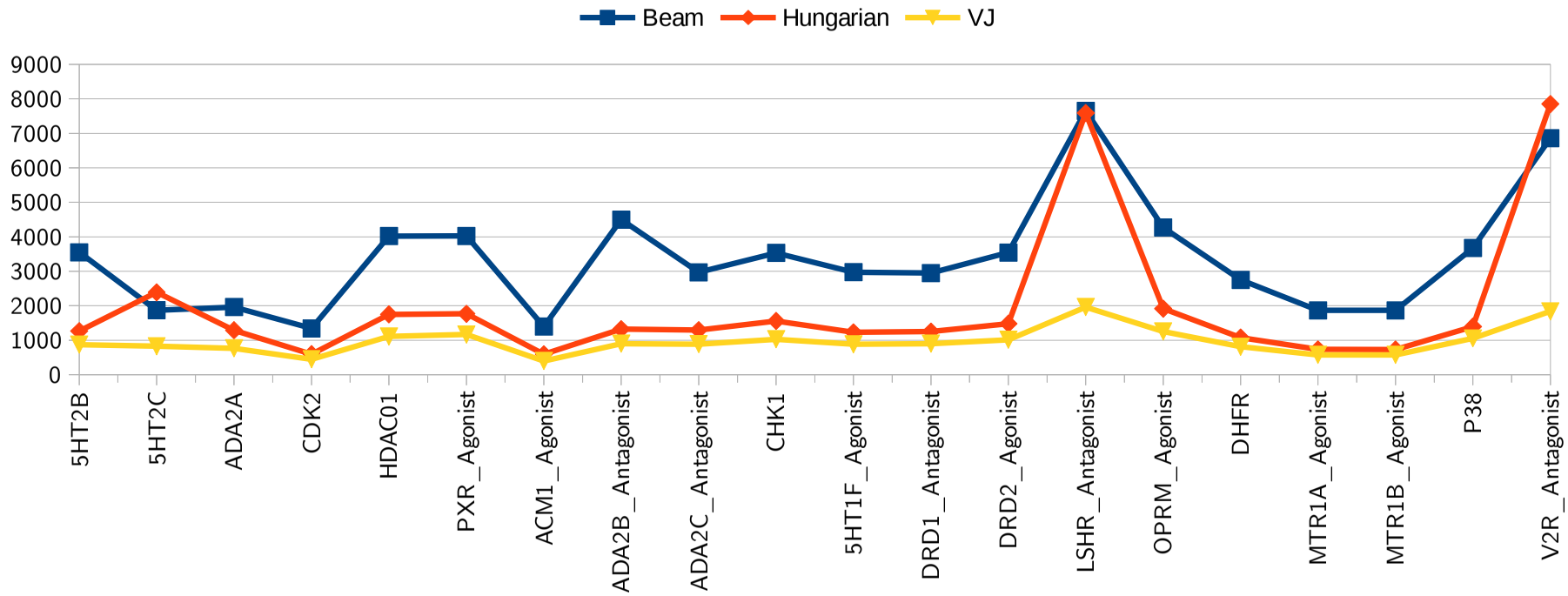
Tabulka 5.7: Srovnání AUC pro různé volby parametru α při volbě atributu elektronegativita a algoritmu Beam.

	eneg-Beam-0.3	eneg-Hungarian-0.3	eneg-VJ-0.3
8.0-8.5			
5HT2B	0.628	0.840	0.574
5HT2C	0.574	0.818	0.553
ADA2A	0.607	0.852	0.562
CDK2	0.550	0.774	0.617
HDAC01	0.488	0.693	0.580
PXR_Agonist	0.583	0.732	0.551
8.5-9.0			
ACM1_Agonist	0.735	0.907	0.776
ADA2B_Antagonist	0.610	0.857	0.615
ADA2C_Antagonist	0.587	0.860	0.586
CHK1	0.611	0.800	0.574
9.0-9.5			
5HT1F_Agonist	0.702	0.941	0.712
DRD1_Antagonist	0.642	0.960	0.626
DRD2_Agonist	0.668	0.913	0.561
LSHR_Antagonist	0.577	0.885	0.559
OPRM_Agonist	0.768	0.961	0.568
9.8-1.0			
DHFR	0.722	0.955	0.763
MTR1A_Agonist	0.902	0.948	0.677
MTR1B_Agonist	0.910	0.938	0.642
P38	0.742	0.916	0.476
V2R_Antagonist	0.770	0.986	0.715

Tabulka 5.8: Srovnání AUC pro různé volby algoritmu při volbě atributu elektronegativita a $\alpha = 0{,}3$.

	valence-Hungarian-0.3	symbol-Hungarian-0.3	eneg-Hungarian-0.3
8.5-9.0			
ACM1_Agonist	0.900	0.919	0.907
ADA2B_Antagonist	0.870	0.869	0.857
ADA2C_Antagonist	0.866	0.870	0.860
CHK1	0.851	0.806	0.800
9.0-9.5			
5HT1F_Agonist	0.935	0.945	0.941
DRD1_Antagonist	0.953	0.967	0.960
DRD2_Agonist	0.895	0.919	0.913
LSHR_Antagonist	0.874	0.892	0.885
OPRM_Agonist	0.952	0.970	0.961
9.8-1.0			
DHFR	0.892	0.961	0.955
MTR1A_Agonist	0.947	0.950	0.948
MTR1B_Agonist	0.953	0.945	0.938

Tabulka 5.9: Srovnání AUC pro různé volby atributů při použití maďarského algoritmu a $\alpha = 0{,}3$.



Obrázek 5.6: Srovnání doby běhu algoritmů Beam, Hungarian a VJ na testovacích datasetech.

	ap	ecfp2	maccs	tt	mcs	ged
8.0-8.5						
5HT2B	0.776	0.816	0.815	0.821	0.820	0.840
5HT2C	0.845	0.858	0.848	0.858	0.804	0.818
ADA2A	0.843	0.888	0.857	0.834	0.822	0.852
CDK2	0.831	0.780	0.800	0.855	0.805	0.774
HDAC01	0.813	0.838	0.764	0.832	0.785	0.693
PXR_Agonist	0.831	0.883	0.868	0.994	0.903	0.732
8.5-9.0						
ACM1_Agonist	0.871	0.900	0.904	0.859	0.851	0.907
ADA2B_Antagonist	0.879	0.916	0.893	0.853	0.845	0.857
ADA2C_Antagonist	0.875	0.918	0.886	0.860	0.859	0.860
CHK1	0.908	0.938	0.856	0.874	0.882	0.800
9.0-9.5						
5HT1F_Agonist	0.941	0.948	0.946	0.961	0.881	0.941
DRD1_Antagonist	0.942	0.948	0.946	0.942	0.875	0.960
DRD2_Agonist	0.948	0.966	0.962	0.936	0.944	0.913
LSHR_Antagonist	0.931	0.939	0.923	0.956	0.919	0.885
OPRM_Agonist	0.951	0.965	0.929	0.955	0.946	0.961
9.8-1.0						
DHFR	0.981	0.999	0.996	0.967	0.986	0.955
MTR1A_Agonist	0.993	0.989	0.981	0.997	0.931	0.948
MTR1B_Agonist	0.994	0.991	0.986	0.989	0.907	0.938
P38	0.980	0.992	0.979	0.991	0.993	0.916
V2R_Antagonist	0.980	0.992	0.982	0.993	0.981	0.986

Tabulka 5.10: Srovnání FP metod, MCS a GED podle AUC.

	ap	tt	maccs	ecfp2	mcs	ged
8.0-8.5	0.005	0.005	0.005	0.005	0.005	0.005
5HT2B	78.720	79.704	90.528	85.608	61.008	68.880
5HT2C	85.608	97.416	99.384	96.432	71.832	79.704
ADA2A	89.544	98.400	103.320	108.240	70.848	75.768
CDK2	69.859	75.762	48.213	78.714	63.955	52.472
HDAC01	46.248	61.008	18.696	47.232	48.216	27.552
PXR_Agonist	127.885	159.103	133.092	145.253	129.837	75.445
8.5 – 9.0						
ACM1_Agonist	75.767	143.201	100.366	89.542	67.895	92.248
ADA2B_Antagonist	93.480	86.590	105.288	110.208	82.656	92.496
ADA2C_Antagonist	82.656	96.432	110.208	97.416	78.720	79.704
CHK1	117.018	98.400	84.565	133.736	128.818	70.801
9.0 – 9.5						
5HT1F_Agonist	138.744	141.599	141.696	131.856	90.528	146.616
DRD1_Antagonist	123.000	129.888	140.712	143.664	84.624	143.664
DRD2_Agonist	128.904	128.904	148.584	146.616	99.384	122.016
LSHR_Antagonist	134.808	142.680	132.840	134.808	130.872	128.904
OPRM_Agonist	124.968	133.824	141.696	151.536	127.920	148.584
9.8 – 1.0						
DHFR	155.145	149.568	148.301	179.844	150.713	106.449
MTR1A_Agonist	169.248	162.380	164.328	146.616	136.776	115.128
MTR1B_Agonist	167.280	154.488	167.280	147.600	130.872	125.952
P38	150.515	140.712	125.717	161.221	173.655	127.188
V2R_Antagonist	163.344	184.008	165.312	183.024	175.152	168.264

Tabulka 5.11: Srovnání FP metod, MCS a GED pro EF s parametrem 0,5 %.

	ap	tt	maccs	ecfp2	mcs	ged
8.0-8.5	0.01	0.01	0.01	0.01	0.01	0.01
5HT2B	41.328	44.280	48.216	48.216	32.472	41.328
5HT2C	49.200	52.644	54.120	51.660	42.312	45.264
ADA2A	49.692	52.152	55.104	59.040	40.344	40.836
CDK2	38.865	43.293	27.550	43.293	36.897	47.295
HDAC01	29.520	37.884	11.316	31.488	31.980	17.712
PXR_Agonist	66.295	90.154	67.904	74.109	71.439	40.535
8.5 – 9.0						
ACM1_Agonist	43.787	74.632	55.595	53.135	41.819	51.255
ADA2B_Antagonist	47.724	48.707	59.532	60.024	46.740	48.708
ADA2C_Antagonist	45.756	52.152	57.072	57.564	44.280	41.820
CHK1	60.967	54.612	50.641	72.767	68.342	39.825
9.0 – 9.5						
5HT1F_Agonist	76.260	77.192	75.768	69.372	56.088	78.720
DRD1_Antagonist	70.356	76.260	72.816	77.244	51.660	78.228
DRD2_Agonist	72.816	74.784	77.736	79.704	53.628	65.436
LSHR_Antagonist	71.832	76.260	70.356	72.816	71.832	67.404
OPRM_Agonist	66.420	71.832	75.276	81.672	69.864	76.752
9.8 – 1.0						
DHFR	93.404	84.132	92.274	95.589	79.677	65.200
MTR1A_Agonist	90.036	94.928	85.116	86.100	72.816	65.436
MTR1B_Agonist	90.036	83.640	89.544	84.132	70.356	68.880
P38	84.745	79.704	72.530	89.677	93.951	70.884
V2R_Antagonist	86.100	93.972	85.608	93.480	89.544	90.528

Tabulka 5.12: Srovnání FP metod, MCS a GED pro EF s parametrem 1 %.

Závěr

Cílem této práce bylo vyzkoušet grafové přístupy ke zjišťování podobnosti molekul a jejich následné použití v LBVS. Povedlo se nám najít a použít existující implementace největšího společného hranového podgrafu a grafové editační vzdálenosti. Zkoumali jsme výsledky metod s různě striktním nastavením jednotlivých parametrů. Nejlepší z nich jsme pak srovnali se stávajícími metodami.

Zjistili jsme, že grafové přístupy dávají výsledky srovnatelné s výsledky metod využívajícími molekulární otisky prstů. Ukazuje se tedy, že absence využití grafových algoritmů v LBVS je způsobená hlavně složitostí výpočtů a nikoli nedostatečnými výsledky. Věříme, že výsledky grafových přístupů je možné dalším výzkumem vylepšit a v rámci možností zrychlit.

Při našem výzkumu jsme narazili na několik oblastí, ve kterých by bylo možné jej rozšířit.

První oblastí je srovnání existujících algoritmů a implementací MCS. Algoritmů i publikovaných implementací existuje mnoho, přesto se nám nepovedlo najít studii, která by je srovnávala v oblasti výkonnosti a kvality.

Dále zůstává mnoho otevřených otázek v oblasti použití GED. Jak jsme zmínili, existuje několik úprav pro speciální typy grafů, pro které existují polynomiální algoritmy. Pokud by se molekulární grafy povedlo převést na stromy nebo rovinné grafy, bylo by pak možné otestovat pro účely LBVS i tyto algoritmy. Tento přístup by mohl přinést zrychlení, které by pro použití v praxi bylo nezbytné. Nevyzkoušená zůstává i přesná varianta grafové editační vzdálenosti.

Seznam použité literatury

[1] Biovia - scientific enterprise software for chemical research. URL http://accelrys.com/.

[2] Creative commons, attribution-sharealike 3.0 unported. https://creativecommons.org/licenses/by-sa/3.0/legalcode. Last retrieved 2017-04-30.

[3] Daylight chemical information systems. URL http://www.daylight.com/.

[4] Indigo cheminformatics library. URL http://ggasoftware.com/opensource/indigo.

[5] Openeye scientific software. URL https://www.eyesopen.com/.

[6] Rdkit: Open-source cheminformatics software. URL http://www.rdkit.org.

[7] (2010). *CTfile Formats*. Symyx Solutions, Inc. URL http://infochim.u-strasbg.fr/recherche/Download/Fragmentor/MDL_SDF.pdf.

[8] Adamson, G. W. a Bush, J. A. (1973). A method for the automatic classification of chemical structures. *Information Storage and Retrieval*, **9**(10), 561 – 568. ISSN 0020-0271. doi: http://dx.doi.org/10.1016/0020-0271(73)90059-4. URL http://www.sciencedirect.com/science/article/pii/0020027173900594.

[9] Adamson, G. W. a Bush, J. A. (1975). A comparison of the performance of some similarity and dissimilarity measures in the automatic classification of chemical structures. *Journal of Chemical Information and Computer Sciences*, **15**(1), 55–58. doi: 10.1021/ci60001a016. URL http://dx.doi.org/10.1021/ci60001a016. PMID: 1127038.

[10] Armitage, J. E. a Lynch, M. F. (1967). Automatic detection of structural similarities among chemical compounds. *J. Chem. Soc. C*, pages 521–528. doi: 10.1039/J39670000521. URL http://dx.doi.org/10.1039/J39670000521.

[11] Bajusz, D., Rácz, A. a Héberger, K. (2015). Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of Cheminformatics*, **7**(1), 20. ISSN 1758-2946. doi: 10.1186/s13321-015-0069-3. URL http://dx.doi.org/10.1186/s13321-015-0069-3.

[12] Bamber, D. (1975). The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, **12**(4), 387 – 415. ISSN 0022-2496. doi: https://doi.org/10.1016/0022-2496(75)90001-2. URL http://www.sciencedirect.com/science/article/pii/0022249675900012.

[13] Bender, A. a Glen, R. C. (2005). A discussion of measures of enrichment in virtual screening: Comparing the information content of descriptors with increasing levels of sophistication. *Journal of Chemical Information and Modeling*, **45**(5), 1369–1375. doi: 10.1021/ci0500177. URL http://dx.doi.org/10.1021/ci0500177. PMID: 16180913.

[14] Bron, C. a Kerbosch, J. (1973). Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, **16**(9), 575–577. ISSN 0001-0782. doi: 10.1145/362342.362367. URL http://doi.acm.org/10.1145/362342.362367.

[15] Brown, R. (1993). *A Hyperstructure Model for Chemical Structure Handling*. University of Sheffield, Sheffield, U.K. URL https://books.google.cz/books?id=r7ADtwAACAAJ. Ph.D. Thesis, Department of Information Studies.

[16] Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, **18**(8), 689 – 694. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/S0167-8655(97)00060-3. URL http://www.sciencedirect.com/science/article/pii/S0167865597000603.

[17] Carhart, R. E., Smith, D. H. a Venkataraghavan, R. (1985). Atom pairs as molecular features in structure-activity studies: definition and applications. *Journal of Chemical Information and Computer Sciences*, **25**(2), 64–73. doi: 10.1021/ci00046a002. URL http://dx.doi.org/10.1021/ci00046a002.

[18] Cereto-Massague, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallve, S. a Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, **71**(C), 58–63.

[19] Clark, R. D. a Roe, D. C. (2010). *Handbook of Chemoinformatics Algorithms*. 1st edition. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-4200-8292-0.

[20] Dalby, A., Nourse, J. G., Hounshell, W. D., Gushurst, A. K. I., Grier, D. L., Leland, B. A. a Laufer, J. (1992). Description of several chemical structure file formats used by computer programs developed at molecular design limited. *Journal of Chemical Information and Computer Sciences*, **32**(3), 244–255. doi: 10.1021/ci00007a012. URL http://dx.doi.org/10.1021/ci00007a012.

[21] Dalke, A. a Hastings, J. (2013). Fmcs: a novel algorithm for the multiple mcs problem. *Journal of Cheminformatics*, **5**(1), O6. ISSN 1758-2946. doi: 10.1186/1758-2946-5-S1-O6. URL http://dx.doi.org/10.1186/1758-2946-5-S1-O6.

[22] Dickinson, P. J., Bunke, H., Dadej, A. a Kraetzl, M. (2003). *On Graphs with Unique Node Labels*, pages 13–23. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-45028-3. doi: 10.1007/3-540-45028-9_2. URL http://dx.doi.org/10.1007/3-540-45028-9_2.

[23] Duesbury, E., Holliday, J. a Willett, P. (2017). Maximum common subgraph isomorphism algorithms. *MATCH Commun. Math. Comput. Chem.*, **77**, 213–232. ISSN 340 - 6253.

[24] Duesbury, E., Holliday, J. a Willett, P. (2017). Maximum common subgraph isomorphism algorithms. *MATCH Commun. Math. Comput. Chem.*, **77**, 229. ISSN 340 - 6253.

[25] Duesbury, E., Holliday, J. a Willett, P. (2015). Maximum common substructure-based data fusion in similarity searching. *Journal of Chemical Information and Modeling*, **55**(2), 222–230. doi: 10.1021/ci5005702. URL http://dx.doi.org/10.1021/ci5005702. PMID: 25602464.

[26] Durand, P. J., Pasari, R., Baker, J. a Tsai, C. (1999). An efficient algorithm for similarity analysis of molecules. *Internet J. Chem.*, **2**, 1–16.

[27] Durant, J. L., Leland, B. A., Henry, D. R. a Nourse, J. G. (2002). Reoptimization of mdl keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, **42**(6), 1273–1280. doi: 10.1021/ci010132r. URL http://dx.doi.org/10.1021/ci010132r. PMID: 12444722.

[28] Egerváry, E. (1931). Matrixok kombinatorius tulajdonságairol. *Matematikai és Fizikai Lapok*, **38**, 16–28.

[29] Engel, T. (2006). Basic overview of chemoinformatics. *Journal of Chemical Information and Modeling*, **46**(6), 2267–2277. doi: 10.1021/ci600234z. URL http://dx.doi.org/10.1021/ci600234z. PMID: 17125169.

[30] Fankhauser, S., Riesen, K. a Bunke, H. (2011). *Speeding Up Graph Edit Distance Computation through Fast Bipartite Matching*, pages 102–111. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-20844-7. doi: 10.1007/978-3-642-20844-7_11. URL http://dx.doi.org/10.1007/978-3-642-20844-7_11.

[31] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, **27**(8), 861 – 874. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.10.010. URL http://www.sciencedirect.com/science/article/pii/S016786550500303X. ROC Analysis in Pattern Recognition.

[32] Fdardel a Commons, W. (2007). Generation of smiles: Break cycles, then write as branches off a main backbone. (ciprofloxacin). URL https://commons.wikimedia.org/w/index.php?curid=2556784. Slight edit by DMacks.

[33] Gao, X., Xiao, B., Tao, D. a Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and Applications*, **13**(1), 113–129. ISSN 1433-755X. doi: 10.1007/s10044-008-0141-y. URL http://dx.doi.org/10.1007/s10044-008-0141-y.

[34] Godden, J. W., Xue, L. a Bajorath, J. (2000). Combinatorial preferences affect molecular similarity/diversity calculations using binary fingerprints and tanimoto coefficients. *Journal of Chemical Information and Computer Sciences*, **40**(1), 163–166. doi: 10.1021/ci990316u. URL `http://dx.doi.org/10.1021/ci990316u`. PMID: 10661563.

[35] Gross, J. L., Yellen, J. a Zhang, P. (2013). *Handbook of Graph Theory, Second Edition*. Chapman and Hall, 2nd edition. ISBN 1439880182, 9781439880180.

[36] Guha, R. (2010). *Handbook of Chemoinformatics Algorithms*. 1st edition. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-4200-8292-0.

[37] Hariharan, R., Janakiraman, A., Nilakantan, R., Singh, B., Varghese, S., Landrum, G. a Schuffenhauer, A. (2011). Multimcs: A fast algorithm for the maximum common substructure problem on multiple molecules. *Journal of Chemical Information and Modeling*, **51**(4), 788–806. doi: 10.1021/ci100297y. URL `http://dx.doi.org/10.1021/ci100297y`. PMID: 21446748.

[38] Hart, P. E., Nilsson, N. J. a Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, **4**(2), 100–107. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

[39] Holt, R. C., Winter, A. a Schurr, A. (2000). Gxl: toward a standard exchange format. In *Proceedings Seventh Working Conference on Reverse Engineering*, pages 162–171. doi: 10.1109/WCRE.2000.891463.

[40] Hull, R. C. a Winter, A. (2000). A short introduction to the gxl software exchange format. In *Proceedings Seventh Working Conference on Reverse Engineering*, pages 299–301. doi: 10.1109/WCRE.2000.891486.

[41] Inc., A. (2011). Maccs structural keys. San Diego, CA.

[42] Jiang, X. a Bunke, H. (1999). Optimal quadratic-time isomorphism of ordered graphs. *Pattern Recognition*, **32**(7), 1273 – 1283. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/S0031-3203(98)00145-9. URL `http://www.sciencedirect.com/science/article/pii/S0031320398001459`.

[43] Jones, W., Chawdhary, A. a King, A. (2015). *Revisiting Volgenant-Jonker for Approximating Graph Edit Distance*, pages 98–107. Springer International Publishing, Cham. ISBN 978-3-319-18224-7. doi: 10.1007/978-3-319-18224-7_10. URL `http://dx.doi.org/10.1007/978-3-319-18224-7_10`.

[44] Jonker, R. a Volgenant, T. (1988). *A shortest augmenting path algorithm for dense and sparse linear assignment problems*, pages 622–622. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-73778-7. doi: 10.1007/978-3-642-73778-7_164. URL `http://dx.doi.org/10.1007/978-3-642-73778-7_164`.

[45] Justice, D. a Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, **28**(8), 1200–1214. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.152. URL http://dx.doi.org/10.1109/TPAMI.2006.152.

[46] Kim, S., Thiessen, P. A., Bolton, E. E., Chen, J., Fu, G., Gindulyte, A., Han, L., He, J., He, S., Shoemaker, B. A., Wang, J., Yu, B., Zhang, J. a Bryant, S. H. (2016). Pubchem substance and compound databases. *Nucleic Acids Research*, **44**(D1), D1202–D1213. doi: 10.1093/nar/gkv951. URL http://dx.doi.org/10.1093/nar/gkv951.

[47] Klebe, G. (2006). Virtual ligand screening: strategies, perspectives and limitations. *Drug Discovery Today*, **11**(13–14), 580 – 594. ISSN 1359-6446. doi: http://doi.org/10.1016/j.drudis.2006.05.012. URL http://www.sciencedirect.com/science/article/pii/S1359644606001784.

[48] Koutra, D., Parikh, A., Ramdas, A. a Xiang, J. (2011). Algorithms for graph similarity and subgraph matching. page 5.

[49] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, **2**(1-2), 83–97. ISSN 1931-9193. doi: 10.1002/nav.3800020109. URL http://dx.doi.org/10.1002/nav.3800020109.

[50] Lipinski, C. A., Lombardo, F., Dominy, B. W. a Feeney, P. J. (1997). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, **23**(1–3), 3 – 25. ISSN 0169-409X. doi: http://doi.org/10.1016/S0169-409X(96)00423-1. URL http://www.sciencedirect.com/science/article/pii/S0169409X96004231. In Vitro Models for Selection of Development Candidates.

[51] McGregor, J. J. (1982). Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, **12**(1), 23. ISSN 1097-024X. doi: 10.1002/spe.4380120103. URL http://dx.doi.org/10.1002/spe.4380120103.

[52] McGregor, J. J. a Willett, P. (1981). Use of a maximum common subgraph algorithm in the automatic identification of ostensible bond changes occurring in chemical reactions. *Journal of Chemical Information and Computer Sciences*, **21**(3), 137. doi: 10.1021/ci00031a005. URL http://dx.doi.org/10.1021/ci00031a005.

[53] Neuhaus, M. a Bunke, H. (2005). Self-organizing maps for learning the edit costs in graph matching. *Trans. Sys. Man Cyber. Part B*, **35**(3), 503–514. ISSN 1083-4419. doi: 10.1109/TSMCB.2005.846635. URL http://dx.doi.org/10.1109/TSMCB.2005.846635.

[54] Neuhaus, M. a Bunke, H. (2004). *An Error-Tolerant Approximate Matching Algorithm for Attributed Planar Graphs and Its Application to Fingerprint Classification*, pages 180–189. Springer Berlin Heidelberg, Berlin,

Heidelberg. ISBN 978-3-540-27868-9. doi: 10.1007/978-3-540-27868-9_18. URL http://dx.doi.org/10.1007/978-3-540-27868-9_18.

[55] Nilakantan, R., Bauman, N. a Dixon, J. S. (1987). Topologial torsion: A new molecular descriptor for sar applications. comparison with other descriptors. *J. Chem. Inf. Comput. Sci.*, **27**(2), 82–85. ISSN 0095-2338. doi: 10.1021/ci00054a008. URL http://dx.doi.org/10.1021/ci00054a008.

[56] O'Boyle, N. M. a Sayle, R. A. (2016). Comparing structural fingerprints using a literature-based similarity benchmark. *Journal of Cheminformatics*, **8**(1), 36. ISSN 1758-2946. doi: 10.1186/s13321-016-0148-0. URL http://dx.doi.org/10.1186/s13321-016-0148-0.

[57] O'Boyle, N. M. a Sayle, R. A. (2016). Which is the best fingerprint for medicinal chemistry? URL https://nextmovesoftware.com/blog/2016/07/07/which-structural-fingerprint-is-best/. 7th Joint Sheffield Conference on Chemoinformatics, poster.

[58] Ostergard, P. R. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, **120**(1–3), 197 – 207. ISSN 0166-218X. doi: http://doi.org/10.1016/S0166-218X(01)00290-6. URL http://www.sciencedirect.com/science/article/pii/S0166218X01002906. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.

[59] Prosser, P. (2012). Exact algorithms for maximum clique: A computational study. *Algorithms*, **5**(4), 545–587. ISSN 1999-4893. doi: 10.3390/a5040545. URL http://dx.doi.org/10.3390/a5040545.

[60] Rahman, S. A., Bashton, M., Holliday, G. L., Schrader, R. a Thornton, J. M. (2009). Small molecule subgraph detector (smsd) toolkit. *Journal of Cheminformatics*, **1**(1), 12. ISSN 1758-2946. doi: 10.1186/1758-2946-1-12. URL http://dx.doi.org/10.1186/1758-2946-1-12.

[61] Raymond, J. W. a Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, **16**(7), 521–533. ISSN 1573-4951. doi: 10.1023/A:1021271615909. URL http://dx.doi.org/10.1023/A:1021271615909.

[62] Raymond, J. W. a Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, **16**(7), 523. ISSN 1573-4951. doi: 10.1023/A:1021271615909. URL http://dx.doi.org/10.1023/A:1021271615909.

[63] Raymond, J. W. a Willett, P. (2002). Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2d chemical structure databases. *Journal of Computer-Aided Molecular Design*, **16**(1), 59–71. ISSN 1573-4951. doi: 10.1023/A:1016387816342. URL http://dx.doi.org/10.1023/A:1016387816342.

[64] Raymond, J. W., Gardiner, E. J. a Willett, P. (2002). Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, **45**, 2002.

[65] Raymond, J. W., Gardiner, E. J. a Willett, P. (2002). Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm. *Journal of Chemical Information and Computer Sciences*, **42**(2), 305–316. doi: 10.1021/ci010381f. URL `http://dx.doi.org/10.1021/ci010381f`. PMID: 11911700.

[66] Riesen, K. (2015). *Structural Pattern Recognition with Graph Edit Distance - Approximation Algorithms and Applications*. Advances in Computer Vision and Pattern Recognition. Springer. ISBN 978-3-319-27251-1. doi: 10.1007/978-3-319-27252-8. URL `http://dx.doi.org/10.1007/978-3-319-27252-8`.

[67] Riesen, K., Neuhaus, M. a Bunke, H. (2007). *Bipartite Graph Matching for Computing the Edit Distance of Graphs*, pages 6 – 7. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-72903-7. doi: 10.1007/978-3-540-72903-7_1. URL `http://dx.doi.org/10.1007/978-3-540-72903-7_1`.

[68] Riesen, K., Emmenegger, S. a Bunke, H. (2013). *A Novel Software Toolkit for Graph Edit Distance Computation*, pages 142–151. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-38221-5. doi: 10.1007/978-3-642-38221-5_15. URL `http://dx.doi.org/10.1007/978-3-642-38221-5_15`.

[69] Riniker, S. a Landrum, G. A. (2013). Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of Cheminformatics*, **5**(1), 26. ISSN 1758-2946. doi: 10.1186/1758-2946-5-26. URL `http://dx.doi.org/10.1186/1758-2946-5-26`.

[70] Rogers, D. a Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, **50**(5), 742–754. doi: 10.1021/ci100050t. URL `http://dx.doi.org/10.1021/ci100050t`. PMID: 20426451.

[71] Rogers, D. a Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, **50**(5), 742–754. doi: 10.1021/ci100050t. URL `http://dx.doi.org/10.1021/ci100050t`. PMID: 20426451.

[72] Sanfeliu, A. a Fu, K. S. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-13**(3), 353–362. ISSN 0018-9472. doi: 10.1109/TSMC.1983.6313167.

[73] Sayle, R., May, J., O'Boyle, N., Grant, J. A., Senger, S. a Green, D. V. (2016). Chemical similarity based on graph edit distance: Efficient implementation and the challenges of evaluation. URL `http://cisrg.`

shef.ac.uk/shef2016/talks/oral13.pdf. 7th Joint Sheffield Conference on Chemoinformatics.

[74] Solé-Ribalta, A., Serratosa, F. a Sanfeliu, A. (2012). On the graph edit distance cost: Properties and applications. *IJPRAI*, **26**(5). doi: 10.1142/S021800141260004X. URL http://dx.doi.org/10.1142/S021800141260004X.

[75] Steinbeck, C., Han, Y., Kuhn, S., Horlacher, O., Luttmann, E. a Willighagen, E. (2003). The chemistry development kit (cdk): An open-source java library for chemo- and bioinformatics. *Journal of Chemical Information and Computer Sciences*, **43**(2), 493–500. doi: 10.1021/ci025584y. URL http://dx.doi.org/10.1021/ci025584. PMID: 12653513.

[76] Swets, J., Dawes, R. a Monahan, J. (2000). Better decisions through science. *Scientific American*, **283**(4), 82–87. ISSN 0036-8733. URL http://ist-socrates.berkeley.edu/~maccoun/LP_SwetsDawesMonahan2000.pdf.

[77] Takahashi, Y., Satoh, Y., Suzuki, H. a ichi Sasaki, S. (1987). Recognition of largest common structural fragment among a variety of chemical structures. *Analytical Sciences*, **3**(1), 23–28. doi: 10.2116/analsci.3.23.

[78] Torsello, A., Hidovic-Rowe, D. a Pelillo, M. (2005). Polynomial-time metrics for attributed trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(7), 1087–1099. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.146.

[79] Truchon, J.-F. a Bayly, C. I. (2007). Evaluating virtual screening methods good and bad metrics for the "early recognition"problem. *Journal of Chemical Information and Modeling*, **47**(2), 488–508. doi: 10.1021/ci600426e. URL http://dx.doi.org/10.1021/ci600426e. PMID: 17288412.

[80] Varkony, T. H., Shiloach, Y. a Smith, D. H. (1979). Computer-assisted examination of chemical compounds for structural similarities. *Journal of Chemical Information and Computer Sciences*, **19**(2), 104–111. doi: 10.1021/ci60018a014. URL http://dx.doi.org/10.1021/ci60018a014.

[81] Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, **28**(1), 31–36. doi: 10.1021/ci00057a005. URL http://dx.doi.org/10.1021/ci00057a005.

[82] Wermuth, C., Ganellin, C., Lindberg, P. a Mitscher, L. (1998). Glossary of terms used in medicinal chemistry (iupac recommendations 1998). *Pure and Applied Chemistry*, **70**, 1140–1141. doi: 10.1351/pac199870051129.

[83] Whitney, H. (1932). Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, **54**(1), 150–168. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2371086.

[84] Zeng, Z., Tung, A. K. H., Wang, J., Feng, J. a Zhou, L. (2009). Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, **2**(1), 25–36. ISSN 2150-8097. doi: 10.14778/1687627.1687631. URL http://dx.doi.org/10.14778/1687627.1687631.

[85] Zhang, K. a Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, **18**(6), 1245–1262. doi: 10.1137/0218082. URL http://dx.doi.org/10.1137/0218082.

[86] Škoda, P. a Hoksza, D. (2016). Benchmarking platform for ligand-based virtual screening. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1220–1227. doi: 10.1109/BIBM.2016.7822693.

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

AP	*Atom Pairs*, atomové páry, viz kap. 1
AP	*Assignment Problem*, viz kap. 3
API	*Application Programming Interface*, rozhraní pro programování aplikací
CXL	formát pro reprezentaci množin grafů založený na xml, viz 4.2.5
FP	*Fingerprints*, molekulární otisky prstů, viz kap. 1
GED	*Graph Edit Distance*, grafová editační vzdálenost, viz kap.3
GMT	*Graph Matching Toolkit*, implementace MCES od Kaspara Riesena[68]
GXL	formát pro reprezentaci grafů založený na xml, viz sekce 4.2.4
HS	*hyperstructure*, viz kap. 2
JSON	*Javascript Object Notation*, viz sekce 4.2.6
LBVS	*Ligand Based Virtual Screening*, virtuální screening založený na ligandech
MACCS	*Molecular ACCess System*[41]
MCES	*Maximum Common Edge Subgraph*, největší společný hranový podgraf, viz definice 4
MCIS	*Maximum Common Induced Subgraph*, největší společný indukovaný podgraf, viz definice 3
MCS	*Maximum Common Subgraph*, největší společný podgraf, viz kapitola 2
MOS	*Maximum Overlapping Set*, maximální překrývající se množina
SBVS	*Structure-Based Virtual Screening*, virtuální screening založený na struktuře
SDF	*(Structure Data File)*, viz sekce 4.2.1
SMILES	*Simplified Molecular-Input Line-Entry System*, způsob reprezentace molekul, viz sekce 4.2.2
SMARTS	*SMiles ARbitrary Target Specification (SMARTS)*, viz sekce 4.2.3
TT	*Topological Torsion*, druh molekulárních otisků prstů, viz kap. 1
VJ	*Volgenant-Jonker*, algoritmus řešící assignment problem, viz sekce 3.2.4

Přílohy

V elektronické příloze k této práci můžeme nalézt

- testovací datasety
  Testovací datasety jsou uložené ve složkách podle průměrné obtížnosti. Složky obsahují soubory ve formátu SDF s reprezentacemi molekul a ve složce `random_00_30_100_20_4900` vždy 10 předgenerovaných náhodných výběrů ve formátu JSON.

- upravenou verzi Graph Matching Toolkit od Kaspara Riesena[68]
  Složka `GraphMatchingToolkit` obsahuje zdrojové kódy v jazyce Java. Požadována je verze 1.7 či vyšší. Námi upravené funkce a třídy jsou označené komentářem.

- konfigurační soubory
  Ve složce `config` jsou k dispozici konfigurační soubory pro spouštění MCS i GED, ve formátu JSON. V této složce je též .prop konfigurační soubor pro GMT.

- skripty
  Skripty pro spouštění MCS i GED podle zadaných parametrů. Hlavní skript je pojmenovaný `main.sh` a vyžaduje jako parametr cestu ke konfiguračnímu souboru. Volitelně je možné zadat název složky, do které se mají umísťovat výstupní soubory. Ostatní skripty a zdrojové kódy jsou popsané v textu práce. Pro fungování skriptů je potřebné mít nainstalovanou knihovnu RDKit a cestu k ní uloženou v proměnné RDBASE.

- výsledky
  Data, která jsme použili pro tvorbu tabulek a grafů v této práci jsou uložena ve složce `results` ve formátu CSV.