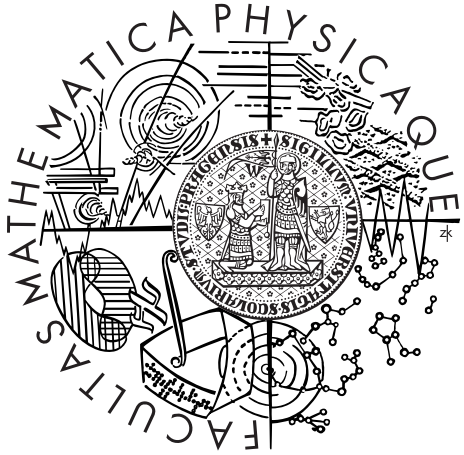
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Marek Mikeš

Explorace chemického prostoru za pomoci scaffold hoppingu

Katedra softwarového inženýrství

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

Studijní program: Informatika

Studijní obor: Softwarové systémy

Praha 2014

Zde bych chtěl poděkovat svému vedoucímu diplomové práce RNDr. Davidovi Hokszovi, Ph.D. za podnětné připomínky a vedení. V neposlední řadě chci poděkovat své přítelkyni Mgr. Barboře Hankové za zázemí, které mi poskytovala, a za její korekturu práce. Nemohu opomenout ani rodinu, která mě po celou dobu podporovala.

Prohlašuji, že jsem tuto diplomovou práci vypracoval 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 v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 31. 7. 2014 Podpis autora

Název práce: Explorace chemického prostoru za pomoci scaffold hoppingu

Autor: Marek Mikeš

Katedra: Katedra softwarového inženýrství

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

Abstrakt: Práce vychází ze SW projektu Molpher, který vznikl jako klient-server aplikace pro hledání cesty v chemickém prostoru mezi dvěmi vstupními molekulami. Cílem diplomové práce bylo rozšířit funkcionalitu o techniku scaffold hopping, kdy je molekula reprezentována zjednodušenou formou (scaffoldem). Bylo potřeba definovat několik úrovní granularity daných scaffoldů a pro každou úroveň morfovací operátory. Serverová část aplikace byla upravena, aby se neporušila silná paralelizace. Součástí práce je i experimentální ověření schopnosti nalezení cesty mezi dvojicí molekul s touto funkcionalitou a bez ní.

Klíčová slova: Scaffold hopping, morf, morfovací operátor

Title: Scaffold hopping-based exploration of chemical space

Author: Marek Mikeš

Department: Department of Software Engineering

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

Abstract: This work is based on the Molpher SW project, which is client-server application aiding exploration of chemical space between two input molecules. Aim of master thesis was modify the current version of program to manage scaffold hopping technique. This technique represents molecule in a simplified way. The simpler molecule is called scaffold. First of all there was need to define several levels of granularity and for each level define morphing operators. Server was modified with respect for parallelization. Experimental exploration of chemical space with and without the new feature is part of this work too.

Keywords: Scaffold hopping, morph, morphing operator

Obsah

Úvod

Tato práce vychází ze SW projektu Molpher obhájeného na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze v roce 2012. Jedná se o multiplatformní aplikaci, jež prohledává prostor chemických sloučenin s cílem nalézt nové sloučeniny, které budou ve formě léků léčit. Jelikož prostor chemických sloučenin je tak velký [16], je zde kladen důraz na rychlost a proto je program silně paralelizovaný. Je rozdělený na klientskou a serverovou část, kde v ideálním případě je serverová část uložena na výkonném počítači (serveru) a klientká část kdekoliv je potřeba.

Prostor chemických sloučenin můžeme přirovnat k vesmíru, oba dva jsou totiž nepředstavitelně velké. Celkový počet malých organických molekul, které existují v chemickém prostoru, byl odhadnut na více než 10^{60} [16]. Je to ohromné číslo vzhledem k množství molekul, které byly dosud objeveny, jelikož zkoumání chemického prostoru bylo značně omezené. Stejně jako je ve vesmíru mnoho prázdna, i prostor chemických sloučenin obsahuje mnoho biologicky nezajímavých látek. Někdy ale (spíše vyjímečně) při notné dávce štěstí se naleznou "hvězdy" v chemickém prostoru – molekuly, které mohou ovlivňovat biologické procesy. Tyto molekuly pak hrají roli při výrobě léků. Podrobné prohledání chemického prostoru je prakticky nemožné vzhledem k jeho velikosti. Klíčová otázka zní, jakým způsobem bychom měli procházet oblasti chemického prostoru, abychom získávali molekuly s chtěnou biologickou aktivitou.

Existují různé online chemické databáze obsahující počítačové reprezentace sloučenin. Tyto databáze ale obsahují pouze malou část chemického prostoru. Například PubChem obsahuje informace o 52.4 miliónech sloučenin [23] a Chemical Abstracts má více než 89.1 miliónů organických a anorganických látek [4] (počty jsou platné k datu 28. 7. 2014). O každé molekule můžeme zjistit její různé vlastnosti jako je počet jejích atomů, molekulová hmotnost, počet cyklů, které obsahuje, počet vazeb, vzdálenosti mezi atomy a tak dále. Tyto informace popisující molekulu se nazývají *deskriptory*. Velký problém spočívá v různosti prostorů chemických sloučenin, protože obvykle bývají definovány různými sadami deskriptorů. Chemické prostory jsou různé, když používají nestejné sady deskriptorů, což znamená, že každý prostor nahlíží na molekuly jinak. Vizualizace chemického prostoru do 2D/3D záleží na použitých deskriptorech i na metodách měřících vzdálenosti mezi molekulami (tzv. podobnostních metodách). Bohužel v tomto oboru zatím nebyla představena obecně platná reprezentace chemických prostorů, která by problém sjednotila a tím i vyřešila [9].

Procházení chemického prostoru je technika umožňující identifikovat v prostoru všech chemických sloučenin cestu, která obsahuje kandidáty na látky mající podobnou aktivitu jako start a cíl této cesty. Při identifikování těch správných kandidátů se molekuly porovnávají s cílovou molekulou na podobnost (pomocí podobnostní metody), kde záleží na struktuře molekul, a nejvyšší šanci na nalezení cesty mají kandidáti, kteří jsou nejvíce podobní cíly. Procházení chemického prostoru záleží na způsobu, jakým jsou zakódovány informace o struktuře a vlastnostech molekuly [39]. Dva hlavní přístupy zakódování informací o molekulách jsou *vektory deskriptorů* a *grafy* (aplikace Molpher používá první přístup s deskriptory):

- *Vektory deskriptorů* – v tomto případě jsou molekuly reprezentovány multi-

dimenzionálním vektorem obsahujícím deskriptory molekuly [37]. Pro vizualizaci těchto multi-dimenzionálních dat jsou potřeba metody, které dimenze redukují. Nejvíce používaná je metoda PCA (Principal Component Analysis)[13] či MDS (Multidimensional Scaling)[30]. Například mapovací systém ChemGPS [24] používá PCA pro vytvoření "navigační mapy" v prostoru léčiv. Další systém založený na PCA, který porovnává množiny sloučenin, se jmenuje DRCS (Delimited Reference Chemical Subspace)[10, 5].

- *Grafy* – v chemickém prostoru založeném na grafech jsou sloučeniny zjednodušeny na takzvané *scaffoldy* [1]. Množiny molekul jsou uspořádány jako scaffoldy v jednoznačném stromu, kde je zachyceno postupné odstraňování kruhů ze složitějších scaffoldů (obr. 3.24)[31]. Tento strom scaffoldů byl úspěšně aplikován na analyzování chemických dat [17, 28] a pro identifikaci nových bioaktivních oblastí chemického prostoru [40].

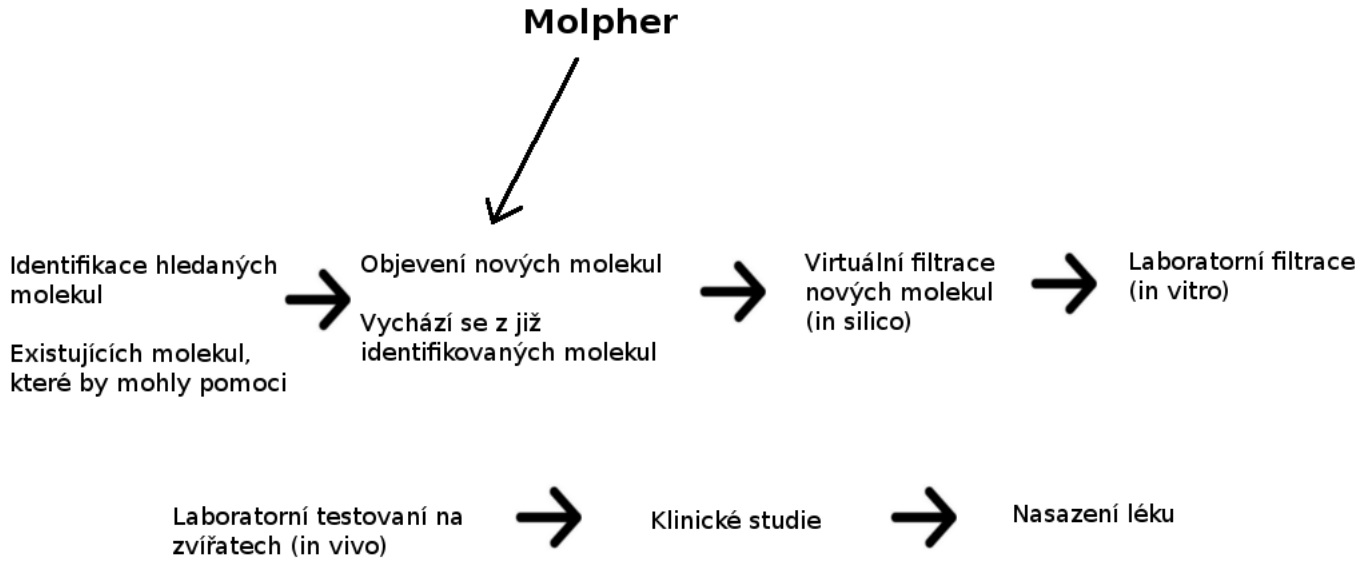
Tato diplomová práce se skládá ze tří kapitol. V první kapitole čtenář zjistí obecné informace o projektu Molpher, ve druhé je obsažen popis nově přidané techniky scaffold hopping [2] a v závěrečné třetí kapitole jsou výsledky experimentů.

1. Projekt Molpher

Jak již bylo zmíněno v úvodu, aplikace Molpher je softwarový projekt vytvořený na MFF UK v Praze na kterém jsem se osobně podílel.

Cílem projektu Molpher bylo vytvoření škálovatelné a interaktivní aplikace, která bude prohledávat chemický prostor sloučenin a ulehčí tak práci pracovníkům v chemickém a lékařském odvětví při odhalování nových látek s definovanými vlastnostmi podobné stávajícím látkám. Tyto látky se mohou uplatnit ve velkém množství případů. Ať už se jedná o léčiva, potravinové doplňky či jiné produkty v chemickém oboru.

Budu-li mluvit o výrobě léčiv, aplikace Molpher je v řetězci mnoha úkonů, které je potřeba udělat, aby byl nový lék zaveden do lékáren, mezi prvními úkony (viz obrázek 1.1). Vydá obsáhlou množinu sloučenin, která obsahuje i neexistující molekuly. Ty následně projdou softwarovou, ale i ruční, filtrací a až poté se případně testují tyto sloučeniny laboratorně. Molpher tedy slouží k nalezení možných nových sloučenin, které jsou něčím podobné již existujícím sloučeninám. Hledají se takové molekuly, které obsahují definované vlastnosti obou zadaných molekul.



Obrázek 1.1: Molpher v procesu hledání léčiv.

Projekt Molpher virtuálně prozkoumává prostor chemických sloučenin za účelem objevení nových, potenciálně užitečných, molekul. Pro systematické objevování prostoru byla navržena metoda "molekulární morfování". Tato metoda byla inspirována efekty v animačních filmech, kde se jeden obrázek postupnými změnami mění v jiný (v angličtině *morph* znamená postupně se měnit). Podobně startovací molekula je změněna na cílovou aplikováním morfovacích operátorů, které dělají jednoduché změny ve struktuře, jako je například přidání/odebrání atomu/vazby. V případě, kdy startovací i cílová molekula patří do nějaké třídy molekul se stejnými vlastnostmi (například jsou aktivní na stejných receptorech), tak molekuly na vzniklé cestě a v jejím okolí reprezentují hledanou množinu, jelikož obsahuje strukturně podobné molekuly zdrojové a cílové molekule. U těchto

molekul z hledané množiny se předpokládá, že budou mít podobné vlastnosti jako zdroj a cíl, což znamená i lepší vlastnosti. Tato množina molekul představuje hodnotné počáteční data pro následné další virtuální filtrování, aby se identifikovaly ty nejaktivnější molekuly. Předpokládaní kandidáti mohou být dále upraveni a jejich biologická aktivita je nakonec testována v laboratořích [14].

Obrázek 1.2: Molpher pomáhá nalézt nové medikamenty[1].

Molpheru funguje tak, že uživatel zadá dvě molekuly (takzvaná *zdrojová* a *cílová* molekula) společně s dalšími parametry a začne výpočet, který se snaží nalézt cestu v chemickém prostoru obsahující podobné molekuly mezi nimi. Další paramatery jsou morfovací operátory, jiné molekuly ke kterým by daná cesta měla konvergovat (tzn. *návnady*), fingerprint, metoda určující podobnost, jakým způsobem se budou molekuly zobrazovat atd. Při výpočtu se generuje takzvaný *strom kandidátů*, což je stromová struktura, kde molekula představuje jeden uzel a potomek molekuly je její morf vytvořený aplikací jednoho morfovacího operátoru. Za běhu může uživatel měnit nastavení výpočtu i prořezávat větve stromu kandidátů, které se podle jeho názoru vydávají špatným směrem. V ideálním případě se posléze nalezne cesta (užitečné informace o průběhu výpočtu a výsledné cestě se uloží na serveru) a výpočet skončí.

Hledání cesty probíhá iterativně. Jeden výpočet Molpheru (hledání cesty) se též nazývá *job*. V každém kroku se nejprve pomocí molekulárního morfování vytvoří nové morfy, některé (nové morfy i stávající molekuly stromu) molekuly se zahodí. Každý morf má referenci na svého otce (molekulu, ze které byl vytvořen molekulárním morfováním) a tím postupně vzniká stromová struktura, tzv. strom kandidátů. Před prvním krokem výpočtu je do stromu kandidátů vložena startovací molekula jako kořen stromu. Na začátku každého kroku se spustí molekulární morfování na všechny listy stromu. Tím se naleznou nové morfy (nové listy), které mají jako svého rodiče nastavený list, ze kterého vznikly. Nové listy se přidají do stromu kandidátů, pokud již v něm nejsou. Dále celý strom projde filtrací – když je rozhodnuto o smazání molekuly ve stromě, smaže se i celý její podstrom. To, zda se stromová molekula smaže záleží na dvou podmínkách. První podmínka je počet kroků, kdy se daná molekula, ani žádná molekula z jejího podstromu, nepřiblížila cíly – čím více kroků nedošlo ke zlepšení, tím je pravděpodobnější, že se hledání pustilo špatnou cestou. Vzdálenosti se počítají ze dvou *fingerprintů*, které představují matematický popis molekul. Počet kroků potřebný pro smazání

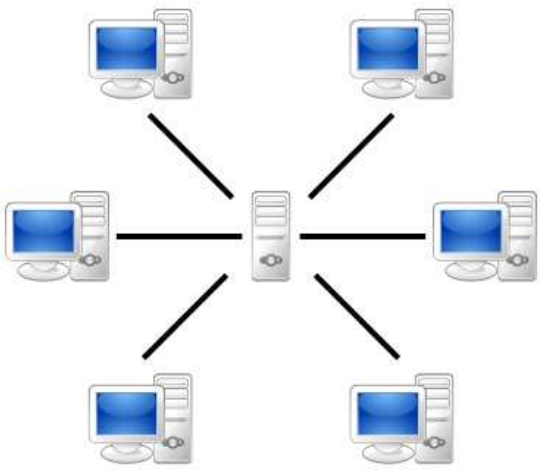
[1]Zdroj: `http://www.equinoxpharma.com/assets/images/molecules_in_drugs.jpg`

stromové molekuly se nastavuje při vytváření jobu. Druhá podmínka zjišťuje, zda uživatel interaktivně prořezal danou molekulu.

Jak již bylo řečeno, Molpher je škálovatelný, což znamená, že se mohou jednoduše přidávat nebo odebírat morfovací operátory, fingerprinty, metody určující podobnost dvou molekul či metody redukující dimenzi chemické sloučeniny do 2D zobrazení.

1.1 Architektura

V ideálním případě by měla být serverová část aplikace na výkonném počítači sama, aby se využilo jejího potenciálu. Samozřejmě je ale možné spustit obě části Molpheru na jednom stroji. Zatímco k serverové části se může připojit mnoho klientů, které mohou případně měnit stav, klientská část může být připojena maximálně k jednomu serveru. Je důležité si uvědomit, že klienti připojení k jednomu serveru nejsou odděleni, protože sdílejí kontext serveru (vidí stejné joby, dostavají stejné zprávy atd.). Komunikace mezi oběmi částmi je asynchronní, takže požadavek klienta je zařazen do fronty, zpracován až na něj přijde čas a následně výstup je oznámen klientovi. Zatímco komunikace klienta je dvoubodová (jeden klient s jedním serverem), server vysílá informace plošně. Vzhledem k tomu, že se mohou přenášet desítky megabytů a k nedostatku zabezpečení komunikace, je vhodné obě komponenty mít na rychlé a zabezpečené lokální síti. Každý uživatel si může vygenerovat cestu která ho zajímá a ostatní uživatelé budou mít právo na čtení nebo i zápis. To je zajištěno zadáváním hesla při změně nastavení každého jobu (heslo je vyžadováno když bylo při vytvoření zadáno).



Obrázek 1.3: Klient-server architektura Molpheru[2].

[2]Zdroj: `http://commons.wikimedia.org/wiki/File:Server-based-network.svg`

Server si ukládá všechny požadavky na vypočtení cest (tzv. joby) a posílá je jeden po druhém do silně paralelizované smyčky. Jakmile je požadavek zařazen do fronty, uživatel může měnit atributy jobu. Dokonce i aktuálně počítaný job může být měněn s určitými výjimkami. Odděleně od samotného výpočtu může serverová část Molpheru generovat okolí i přepočítávat souřadnice (pro zobrazení) vybraných molekul. Co se týká paralelizace, server se skládá ze tří hlavních vláken – první slouží pro komunikaci a zachytává požadavky od klientů, druhé přijímá a počítá joby, a nakonec třetí vlákno generuje okolí nebo přepočítává souřadnice. Obě výpočetní vlákna (druhé a třetí zmíněné) jsou dále rozděleny na další podle možností počítače nebo nastavení uživatelem z příkazové řádky.

Budu-li mluvit o klientské části, skládá se ze dvou vláken – první komunikuje se serverem a druhé vlákno se stará o GUI. Samotné GUI se skládá z widgetu reprezentujícího frontu jobů ze serveru, dialogů pro práci s joby a dalších částí pro snadnou práci s molekulami. Každý job může být otevřen v samostatné záložce, kde uživatel může zkoumat reálný stav výpočtu a případně měnit parametry procházení chemického prostoru. Tento prostor je zobrazen do 2D Eukleidovského prostoru jako graf, přesněji strom. Jednotlivé hrany představují chemické operátory Molpheru. V neposlední řady je také nutné zmínit skutečnost, že klient obsahuje funkcionalitu umožňující dotazovat se online databází chemických molekul, zda obsahují molekulu z jobu.

1.2 Výběr technologií

Pro klientskou část aplikace byla vybrána knihovna Qt [25], která je vyvíjena v programovacím jazyce C++. Jedná se moderní knihovnu určenou pro desktopové aplikace s výbornou dokumentací.

Jelikož se jedná o aplikaci klient-server, bylo nutné zvolit nějaký komunikační prostředek. Bylo nutné nalézt vhodnou middleware technologii, která by splňovala zadané podmínky. Těmi byli GPL kompatibilní licence, stabilní a otestovaná implementace a podpora C++ s dobrou podporou serializace netriviálních datových struktur. Vyvstala otázka, zda k tomuto účelu použít technologii CORBA (Common Object Request Broker Architecture)[6] nebo nějakou jinou méně komplexní alternativu. Bylo rozhodnuto, že CORBA je pro tyto účely až zbytečně složitá a proto nakonec padla volba na RCF (Remote Call Framework)[26]. Stalo se tak i pro to, že interně používá Boost serializaci [3], což znamená, že by nemuselo docházet ke zpomalování během konverze dat.

Dále se řešila otázka výběru vhodné technologie pro paralelizaci na multiprocesorech. V úvahu připadaly dvě možnosti – OpenMP (Open Multi-Processing)[21] nebo TBB (Threading Building Blocks)[36]. Nakonec byla vybrána druhá varianta pro její lepší zpracování a kvůli podpoře pokročilé paralelizace a vhodným datovým kontejnerům.

1.3 Požadavky

Následující vlastnosti by měl systém, na kterém má být Molpher spuštěn, splňovat:

- **Procesor** – jak klientská, tak serverová část programu jsou přeloženy pro procesory s instrukční sadou IA32 (Intel Architecture, 32-bit) nebo AMD64. Navíc server potřebuje vícejádrový procesor s podporou SSE2 (Streaming SIMD Extensions 2). Procesory bez SSE2 nejsou podporovány. Čím více jader daný procesor má, tím je rychlejší díky jeho paralelnímu navržení. Minimální požadavky na klientskou část je jednojádrový procesor s podporou SSE a MMX.

- **Paměť** – obě části Molpheru potřebují desítky megabytů operační paměti při defaultním nastavení, což v dnešní době není žádný problém. Nicméně záleží na nastavení parametrů výpočtu. Při generování vyšších počtů kandidátských molekul a minimálním prořezávání stromu kandidátů může spotřeba paměti vzrůst až o několik řádů.

- **Pevný disk** – s defaultním nastavením parametrů výpočtu zabírá průměrný krok výpočtu (SNP soubor) jednotky megabytů. V případě dlouhých výpočtů, kdy se jedná o stovky iterací, může celý výpočet mít velikost gigabytů. Proto je doporučeno volné místo na disku v řádech desítek gigabytů.

- **Operační systém** – Molpher je aktuálně šířen pro operační systém Windows. Podporovány jsou verze XP a vyšší (32-b i 64-b). Protože se jedná o multiplatformní aplikaci, je zde i možnost kompilace pro Linux.

Instalace aplikace je přímočará. Jelikož byl program hlavně staticky přeložený a zbylé závislosti jsou přiložené v balíčku, stačí jen stažený komprimovaný soubor rozbalit a spustit Molpher. Jediné, co musí být správně nastaveno, je právo zápisu v aktuálním adresáři.

1.4 Morfování

1.4.1 Výběr knihovny

Úplně první verze Molpheru byla založena na sadě chemických nástrojů Open Babel [20]. Pak se začal, v rámci SW projektu, přepisovat. Před přepisováním se zvažovaly všechny varianty, aby byly vybrány ty nejlepší nástroje pro Molpher. Bylo zjištěno, že kromě Open Babel jsou ve hře další dvě možnosti, které splňují požadavky na funkčnost, rozhraní i licenční podmínky – Indigo [12] a RDKit [29].

Všechny tyto tři varianty byly porovnávány pro podporu více vláken, na fingerprinty, podobnostní metody a podporu renderování. Na první pohled vypadalo Indigo jako správná volba kvůli kvalitní dokumentaci, při bližším zkoumání však bylo zjištěno, že nevyhovuje pro paralelní použití. Další jeho problém bylo jeho C rozhraní, takže použití z C++ kódu by bylo méně příjemné. Oproti tomu RDKit poskytuje podporu více vláken, C++ API (Application Programming Interface, tedy rozhraní) a lepší podporu firngerprintů a podobnostních metod oproti Open Babel i Indigo. Co se týče renderování obrázků molekul, RDKit i Indigo umějí pouze spočítat souřadnice atomů a vazeb, ale samotné generování obrázků musí být zajištěno jiným programem. Open Babel nepodporoje vůbec renderování. I když dokumentace RDKit knihovny je spíše podprůměrná, byla vybrána jako nejlepší možnost.

1.4.2 Výpočet

Molpher je navržen s ohledem na paralelizaci i komunikaci po síťi. Níže je vidět seznam datových struktur, které používá jak klientská, tak serverová část Molpheru. Tyto struktury byly navrženy tak, aby podporovaly serializaci, což je nezbytný požadavek pro ukládání stavu výpočtu a kominikaci mezi klientem a serverem.

Dále v textu se bude mluvit o řetězci SMILES (Simplified Molecular-Input Line-Entry System)[32]. Česky se zkratka překládá jako "zjednodušený molekulární systém pro vstupní řádky". Tento systém jednoznačně popisuje jakoukoliv molekulu pomocí ASCII znaků. Je podporován širokou škálou programů používaných v chemickém oboru.

- **MolpherMolecule** obsahuje SMILES řetězec identifikující molekulu, informace o napojení ve stromu kandidátů (SMILES řetezec otce a množinu SMILES řetezců obsahujících potomky), kvalitativní ukazatele (vzdálenost k cíli atd.) a souřadnice molekuly pro zobrazení na klientu. Tato struktura slouží hlavně pro komunikaci mezi oběmi částmi Molpheru, při výpočtu na serveru se konvertuje na RDKit molekulu.
- **MolpherParam** obsahuje různé nastavení výpočtu, které může uživatel zvolit.
- **IterationSnapshot** obsahuje jednu iteraci výpočtu (celý popis jednoho kroku). To znamená všechny informace nastavené uživatelem (zdroj, cíl, parametry, různé metody aj.) a aktuální stav výpočtu (především strom kandidátů, ale i celkový čas výpočtu atd.). Libovolná struktura *IterationSnapshot* uložená jako SNP soubor může být kdykoliv načtena a spuštěna z aktuálního stavu. Uživatelem vytvořený job představuje *IterationSnapshot* s číslem nula a pak každý další krok je o jedna větší.

Molpher podporuje i molekuly, které ovlivňují generování cesty (tzv. návnady). Tato možnost vznikla až v této druhé verzi Molpheru, první verze tuto funkcionalitu neměla. Každý morf je hodnocen podle vzdálenosti k nejbližší návnadě a cílové molekule. Porovnávají se součty těchto dvou vzdáleností a v případě, že se hodnoty rovnají, porovnají se jen vzdálenosti k cíli. Návnady se používají v případech, kdy chceme nalézt cestu s tím, aby výsledné molekuly byly podobné daným návnadám.

1.4.3 Morfovací moduly

V programu existují tři základní třídy modulů – fingerprinty, podobnostní metody a morfovací operátory. Jak již bylo zmíněno, jednotlivé moduly lze jednoduše odebírat či naopak přidávat díky návrhovému vzoru *Strategy* [33]. Následující tři sekce obsahují důležité informace popisující jednotlivé moduly.

Fingerprinty

RDKit poskytuje Morgan fingerprint[3]. Tento typ je obzvláště zajímavý kvůli jeho vlastnostem a rychlému generování. Kvalita Morgan fingerprintu byla porovná-

[3]Více viz `http://tech.knime.org/forum/rdkit/about-circular-fingerprints`.

vána s ostatními fingerprinty a tato metoda je až o řády rychlejší než ostatní. I proto je to defaultní fingerprint Molpheru.

Často bývá výhodnější rozšířit stávající fingerprinty o dodatečné statistiky. Molpher podporuje obě varianty (jak klasickou, tak rozšířenou). V případě rozšířeného fingerprintu se ke klasické variantě přidá navíc vektor bitů. Tento vektor obsahuje počty různých typů atomů v molekule, počty různých typů vazeb (jednoduchá, dvojitá, atd.) v molekule a nakonec počty vazeb mezi různými atomy.

I když RDKit fingerprinty mohou být širokou škálou parametrů nastavovány, Molpher je nechává v defaultním stavu a spoléhá na to, že mají rozumné hodnoty. Kdyby se v budoucnu měli jakýmkoliv způsobem lišit, je na zvážení zda by se měly na tvrdo staticky vložit do kódu nebo by měly být přístupné uživateli v GUI.

Podobnostní metody

Tyto metody obvykle vracejí reálná čísla z uzavřených intervalů [0,1] nebo [-1,1]. Molpher používá tyto podobnostní metody k určení vzdáleností mezi molekulami, jež jsou z intervalu [0,1] – platí pravidlo, že čím menší je vzdálenost, tím větší je podobnost. Každá jednotlivá strategie převádí podobnostní koeficient na vzdálenost.

Většina podobnostních metod nepotřebuje žádné parametry, ale pro metodu Tversky [38] musí být nastaveny dva hmotnostní parametry (jeden pro každou molekulu).

Morfovací operátory

RDKit používá k sanitizaci přísná pravidla, která obsahují více způsobů "vyčištění" molekuly a následně kontrolují validitu (zda jsou molekuly stále chemicky platné). Zatímco samotná sanitizace není nutnou podmínkou pro algoritmus Molpheru a může na ní být nahlíženo jako na zbytečné zpomalování výkonu, většina RDKit metod nedokáže pracovat s molekulami, které tímto procesem neprošly – hrozí pád aplikace, zacyklení nebo vyvolání výjimky. Během vývoje ale bylo zjištěno, že je nezbytná pouze určitá podmnožina sanitizačních činností pro správné fungování. To platí ale pod podmínkou, že molekuly neobsahují informace o aromaticitě. Proto jsou molekuly během celého výpočtu kekulizované.

Knihovna RDKit není úplně přímočará při editacích molekul (přidání atomu, smazání vazby apod.), ale chová se předvídatelným způsobem. Nicméně je zde ještě jeden problém a to, že po smazání vazby z kekulizované a kruh obsahují molekuly skončí sanitizace neúspěchem, i když výsledná molekula dává chemicky význam. Důvodem tohoto chování je pravděpodobně rozbitý nebo nekonzistentní privátní stav RDKit molekuly po dané modifikaci (není jednoznačné zda se jedná o chybu nebo korektní chování). Tento problém je prozatím vyřešen upraveným copy kontruktorem, který zkopíruje molekuly jako graf (množinu atomů a vazeb mezi nimi) a nezkopíruje žádné další dodatečné informace.

Zatímco algoritmus Molpheru se nezajímá o elektrické nabití atomů, sanitizační metoda RDKitu zahazuje molekuly, které nemají chemický význam bez náboje (např. oxid dusnatý). Aby se tomuto zahazování morfů zabránilo, datová struktura *MolpherAtom* obsahuje dodatečné informace. Kromě atomického čísla atomu má v sobě tato struktura i informaci o nabití a váze. Takže když molekula

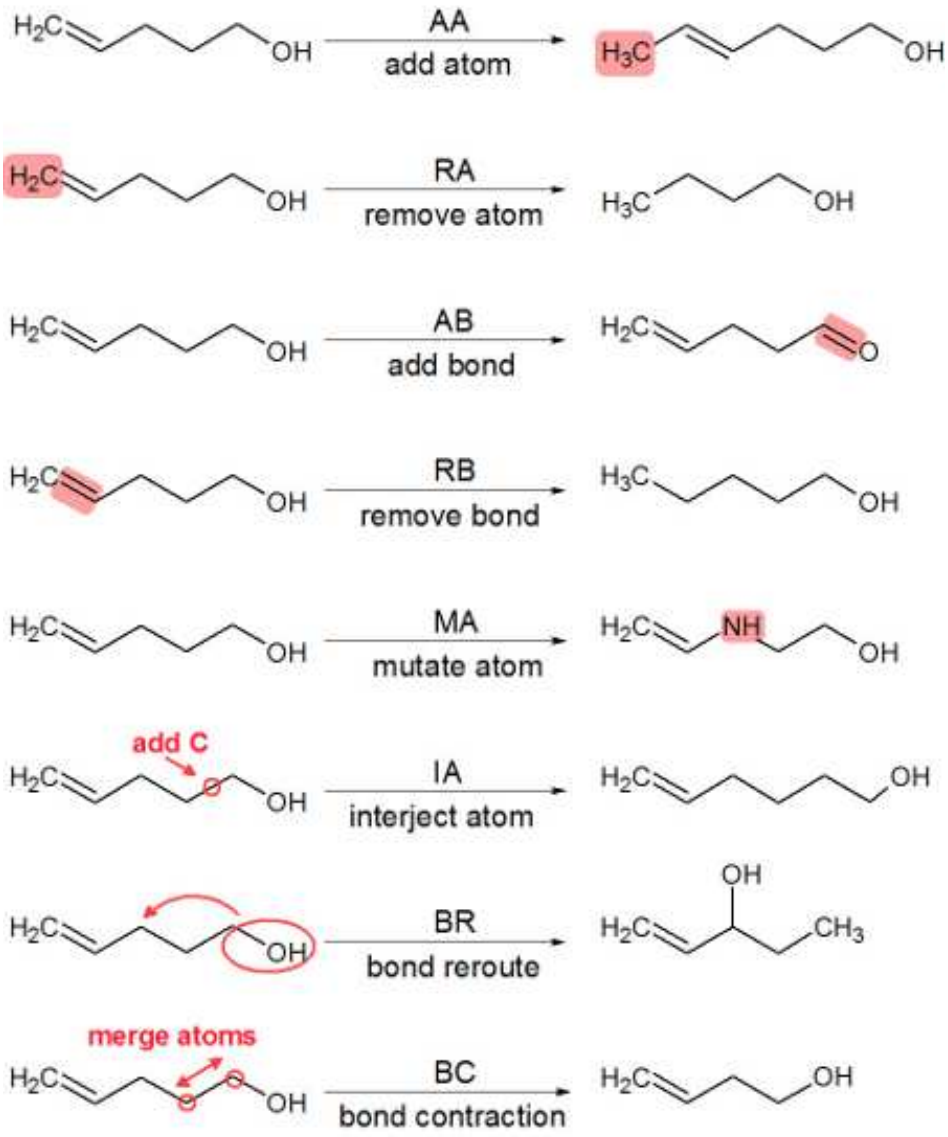
obsahuje neutrální kyslík a aniont kyslíku, funkce *GetAtomTypesFromMol* vrátí obě *MolpherAtom* instance.

Dále popíši všechny morfovací operátory, které Molpher používá. Protože v další části této diplomové práce se bude o nich hojně mluvit, je to vhodné uvést pro dobrou orientaci v textu. Níže uvedené operátory se používaly jak v původní verzi Molpheru, tak se používají v té aktuální s technikou scaffold hopping. Aplikují se na molekulu, tu modifikují a výsledkem je její změněná podoba. Jedná se o následujících osm operátorů:

- *Add Atom* (zkratka AA) – ke vstupní molekule se přidá náhodný atom vyskytující se v cílové molekule a tento atom se spojí s náhodným atomem vstupní molekuly jednoduchou vazbou.
- *Add Bond* (zkratka AB) – přidá se vazba mezi náhodně vybrané dva atomy. V případě, že vazba již existuje, zvýší se její násobnost o jedna.
- *Bond Contraction* (zkratka BC) – smaže se náhodná vazba spojující dva stejné atomy. Tyto dva stejné atomy jsou spojeny do jednoho, tudíž výsledný morf obsahuje o jeden atom méně než vstupní molekula.
- *Bond Reroute* (zkratka BR) – operátor přepojí náhodný konec náhodné vazby tak, že bude spojen s jiným náhodným atomem. Molekula se nesmí při této operaci rozdělit.
- *Interlay Atom* (zkratka IA) – náhodný atom vyskytující se v cílové molekule se vloží mezi dva náhodné atomy vstupní molekuly, které jsou spojeny vazbou. Tedy vazba ve vstupní molekule je rozdělena na dvě.
- *Mutate Atom* (zkratka MA) – náhodný atom ve vstupní molekule je nahrazen náhodným atomem z cílové molekuly.
- *Remove Atom* (zkratka RA) – je odstraněn náhodný atom vstupní molekuly stupně jedna.
- *Remove Bond* (zkratka RB) – násobnost nahodné vazby je snížena o jedna. Když vybraná vazba již je jednoduchá, smaže se. Vykonání tohoto operátoru nesmí způsobit rozpadnutí molekuly na dvě.

Použití jednotlivých morfovacích operátorů je vidět na obrázku 1.4. Červeně jsou zvýrazněny části jednoduchých molekul, které se změnily.

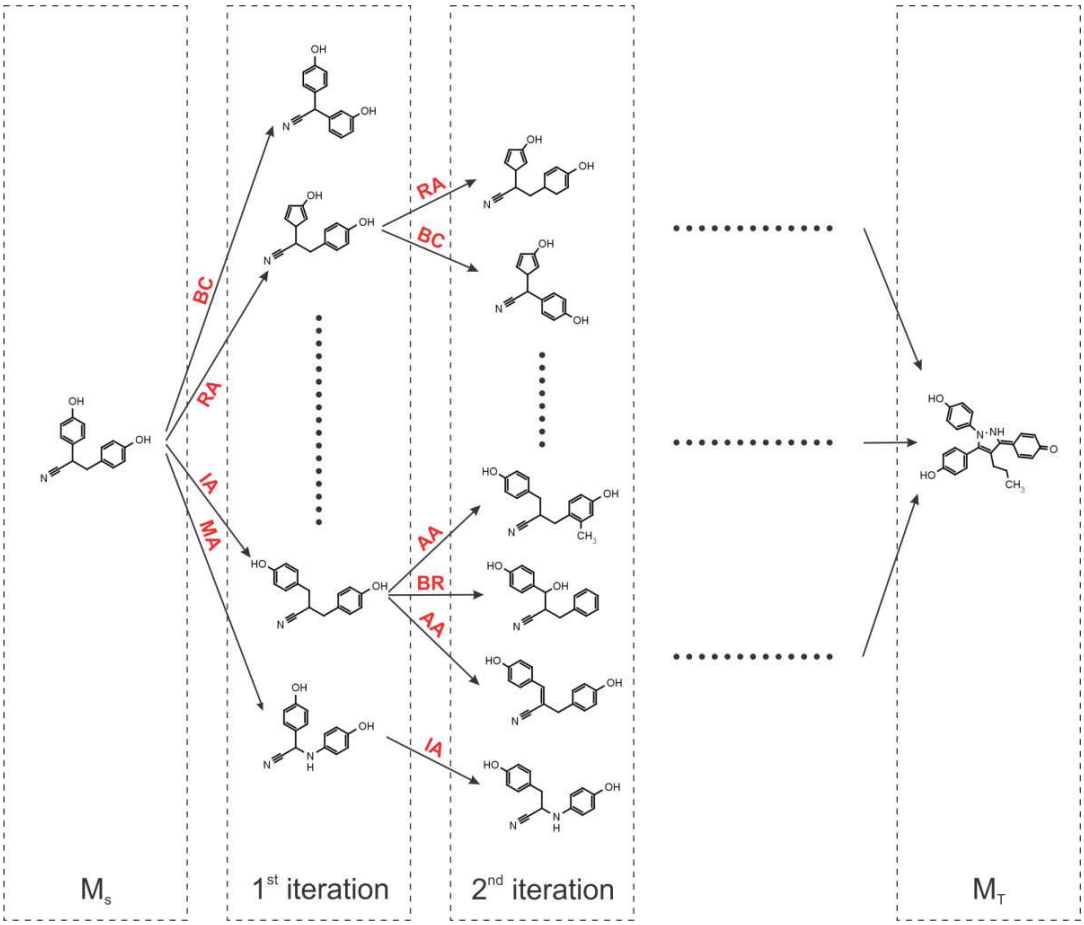
Na dalším obrázku číslo 1.5 je znázorněn princip generování cesty aplikací Molpher. Sloupeček M_S označuje takzvanou nultou iteraci výpočtu, tedy samotnou zdrojovou (startovací) molekulu. Po první iteraci se strom rozrůstá o nové listy, které vznikají aplikací morfovacích (dá se říci i chemických) operátorů na zdrojovou molekulu. V dalších iteracích znovu vznikají nové listy z aktuálně existujících listů až se nakonec nalezne cílová molekula (poslední sloupeček M_T).



Obrázek 1.4: Příklady použití morfovacích operátorů.

1.4.4 Generování okolí

O generování okolí se stará třída *NeighborhoodGenerator*, která pracuje ve vlastním vlákně. Tento koncept byl navržen z toho důvodu, aby uživatel mohl generovat okolí nebo přepočítávat souřadnice, i když se generuje cesta. Okolí i změněné souřadnice nejsou součástí struktury *IterationSnapshot* a tudíž nejsou zahrnuté do historie. Doporučený postup generování okolí je takový, že by se současně měly přepočítat i blízké molekuly z dané iterace a následně bude možné v GUI srovnat blízkosti jak vygenerovaného okolí, tak obklopených molekul. Jelikož přepočítávání souřadnic může být výpočetně náročné (metoda *Kamada-Kawai* [15]), je na zvážení uživatele zda s vygenerováním okolí přepočítá souřadnice všech molekul, nějaké části nebo žádných. Přepočítávání souřadnic a generování okolí mohou fungovat odděleně a tudíž uživatel, když si chce nějakou již existující množinu molekul jinak zobrazit, může přepočítat souřadnice bez generování jakéhokoliv okolí.



Obrázek 1.5: Princip morfování.

Informace o vygenerovaném okolí jsou uloženy v datové struktuře *NeighborhoodTask*. Každá žádost je jednoznačně identifikováná číselnou hodnotou představující čas v mikrosekundách. Samotné okolí je definováno zdrojovou (neboli počáteční) molekulou, maximálním velikostí okolí, maximálním poloměrem co se týče vzdálenosti a maximální hloubkou v kontextu s morfovacími operátory (kolik hran nejdále může být okolní molekula od zadané počáteční). *NeighborhoodGenerator* se iterativně pokouší generovat morfy ze zadané počáteční molekuly nebo již schválených morfů a určuje na základě zadaných omezení, zda patří do okolí či nikoliv. Nemusí se vygenerovat žádná okolní molekula v případě, kdy uživatel zadal parametry příliš striktně. Když počáteční molekula není vyplněna, budou se přepočítávat jen souřadnice zadaných molekul.

1.4.5 Objekty

Na vyšší úrovni existují v Molpheru dvě základní třídy – *PathFinder*, která implementuje základní funkcionalitu, a *NeighborhoodGenerator*, která generuje okolí nebo přepočítává souřadnice na požádání. Obě tyto třídy v určité fázi volají metodu *GenerateMorphs* kvůli získání nových molekul. Obě běží ve svém vlákně a skládají se z několika paralelních funktorů, které jsou vyvolány metodou *operator()*. Zmíněná dvě vlákna představují z pohledu paralelizace základní rozdělení a následně jsou dělena na další TBB vlákna. V případně nečinnosti serverové

části jsou tyto dvě základní vlákna zablokována. Třída *PathFinder* používá ke své práci strukturu *PathFinderContext*, což je jen zkopírovaná struktura *IterationSnapshot*, ale využívající datové struktury podporující konkurenční přístup.

Jak již bylo zmíněno, důležité jsou třídy založené na návrhovém vzoru *Strategy*, které implementují různé fingerprint a podobnostní metody. Fingerprint metody jsou odvozené od třídy *FingerprintStrategy* a podobnostní metody od *SimCoefStrategy*. Pro zjednodušení výpočtů fingerprintů a podobnostních koeficientů za pomoci zmíněných tříd existuje v Molpheru pomocná třída *SimCoefCalculator*.

Dále na nejnižší úrovni v hierarchii objektů existuje další představitel návrhovém vzoru *Strategy* a to třída *MorphingStrategy* představující chemické operátory. Tyto operátory, které se používají na konkrétní molekuly, spolu sdílejí určitá read-only data (*MorphingData*), která jsou předpřipravena z molekuly ze které budou vznikat nové morfy. Pro každý morfovací operátor obsahuje struktura *MorphingData* specifické kandidáty (atomy/vazby), které jsou případně použity při modifikaci molekuly na její morf. Tyto kandidáti odpovídají pravidlům, aby se modifikací molekula nerozpadla na dvě sloučeniny a odpovídala definicím operátorů. Datové struktury jednotlivých modifikačních kandidátů se liší pro každý operátor podle toho, co dělají.

Výše zmíněné třídy jsou volány paralelními metodami *CalculateMorphs*, *CalculateDistances* a *ReturnResults*, které jsou zase volány metodou *GenerateMorphs*. Tato funkce je důležitá v tom smyslu, že odděluje nízkoúrovňový kód generující morfy, a využívající určité chemické knihovny, od vyšší úrovně implementace Molpheru, která většinou nepoužívá tyto knihovny a pracuje s molekulami jako se SMILES řetězci. Jak již název napovídá, úkolem metody *GenerateMorphs* je vracet určitý počet morfů vytvořených aplikací nějakého operátoru na zadanou molekulu, takže volající je může dále filtrovat.

1.4.6 Paralelizace

Jelikož Molpher provádí náročné výpočetní operace, již od počátku vývoje v rámci softwarového projektu bylo jasné, že má potenciál pro masivní paralelizaci na počítačových clusterech. Navržení Molpheru v rámci SW projektu proběhlo s ohledem na paralelizaci pomocí clusterů, nicméně samotná implementace se provedla pro jeden vícejádrový počítač.

Při zvažování výběru technologie pro paralelizaci na multiprocesorech připadaly v úvahu dvě možnosti – OpenMP a TBB. Zatímco OpenMP je přímo součástí kompilátoru a je jednoduché, TBB byla vybrána pro její lepší zpracování a kvůli podpoře pokročilé paralelizace a vhodným datovým kontejnerům. V případě cluster paralelizace, na výběr byly dvě nejznámější MPI (Message Passing Interface) implementace – MPICH (MPI over CHameleon)[19] a Open MPI [22]. Prozkoumáním obou variant se zjistilo, že MPICH je více portabilní a stabilnější implementace, takže mu byla dána přednost před OpenMPI. Tento průzkum byl proveden pro budoucí případné rozšíření na cluster. Za zmínku stojí skutečnost, že standard MPI poskytuje C API, což neni úplně ideální pro implementaci Molpheru. Naštěstí ale zároveň existuje C++ wrapper pro MPI, který je součástí knihovny Boost. Tento wrapper nabízí C++ API, které obsahuje podporu pro STL (Standard Template Library) a uživatelsky definované datové typy. Neměly by vyvstat žádné problémy s kompatibilitou, jelikož MPI od Boostu beží nad

MPICH i Open MPI. Důležitý poznatek pro budoucí přechod na MPI byl, že MPI a TBB se vzájemně nevylučují – až by Molpher fungoval paralelně na počítačových clusterech, tak by například fungování na multiprocesoru bylo stejně zajištěno knihovnou TBB jako tomu teď v aktuální verzi i s technikou scaffold hopping.

Následující popis algoritmu Molpheru se zaměřuje na datové struktury a části kódu, které jsou důležité pro paralelizaci. Tato analýza má za účel dát čtenáři přehled o paralelizačních aspektech celého projektu.

I když to není na první pohled jasné, Molpher velmi často volá generátor náhodných čísel. STL generátor náhodných čísel není vhodný při více vláknech, tudíž zde byla potřeba pro explicitní synchronizaci kvůli vyloučení race condition. Použití explicitního zamykání při generování náhodných čísel by bylo úzkým hrdlem celého programu na nízké úrovni. Řešením tohoto problému je nezávislý generátor náhodných čísel pro každé vlákno zvlášť. V kódu je definován generátor náhodných čísel jako singleton, který je bezpečný pro více vláken a má proměnné uložené v TLS (Thread-local storage) každého vlákna, což poskytuje stejný komfort jako STL. Takže pro každé vlákno je nutná explicitní synchronizace pouze jedinkrát a pak už se mohou generovat náhodná čísla současně.

Na následujících řádcích bude ukázán jeden krok smyčky algoritmu na nejvyšší úrovni pomocí pseudokódu, aby byl vidět způsob fungování Molpheru.

1. Nalezení listů kandidátského stromu.
2. Vygenerování morfů pro každý list stromu.
   (a) Příprava sdílených datových struktur pro generování morfů.
   (b) Vytvoření určitého počtu morfů.
      i. Vybrání náhodného chemického operátoru.
      ii. Aplikace operátoru na daný list.
      iii. Zjištění fingerprintu a vzdálenosti k cíli.
3. Setřídění morfů.
4. Kontrola morfů, zda se nalezla cílová molekula.
5. Filtrace morfů.
6. Vložení přeživších morfů do stromu kandidátů a jeho prořezání.

Body 2 až 5 je možné v budoucnu paralelizovat na počítačových clusterech a body 1 až 6 společne s 2-a a 2-b mohou být paralelizované na vícejádrových procesorech se sdílenou pamětí.

Následující seznam datových struktur je důležitý pro paralelizaci:

- strom kandidátů (*PathFinderContext::candidates*)
  - obsahuje morfy, které přežívají ve výpočtu
  - technicky se nejedná o strom (na propojení se nepoužívají ukazatelé, nýbrž jednoznačné řetězce)
  - globální datová struktura

  - z listů stromu vznikají další molekuly na cestě
  - současný přístup je možný jen na vícejádrovém procesoru nebo masteru uzlu clusteru
  - implementován jako kontejner typu mapa podporující souběžný přístup

- mapa počtu morfů (*PathFinderContext::morphDerivations*)
  - ukládá počty nových morfů pro každý nový morf (zahrnuje neúspěšné i vyfiltrované morfy)
  - globální datová struktura
  - současný přístup (zápis/čtení) možný na celém clusteru
  - implementován jako kontejner typu mapa podporující souběžný přístup
  - je očekáváno rozkopírování na všechny uzly clusteru a následné spojení do master uzlu

- množina přeživších morfů (*MolpherMolecule::historicDescendants*)
  - obsahuje identifikátory morfů, které byly vytvořeny z nějakého listu a přežily alespoň jednu iteraci (pak mohly být prořezáni)
  - každá molekula stromu obsahuje vlastní množinu

- seznam nových morfů jedné iterace (*PathFinder::MoleculeVector*)
  - výstup bodu 2 z výše uvedeného pseudokódu
  - v každé iteraci je tato datová struktura vytvořena a smazána
  - současný přístup je možný jen na vícejádrovém procesoru
  - implementován jako kontejner typu vektor podporující souběžný přístup
  - předpokládané použití všemi uzly clusteru v bodech 3 až 5 výše uvedeného pseudokódu

- pomocná datová struktura pro morfovací operátory (*MorphingData*)
  - obsahuje předpřipravená data pro krok 2-b pseudokódu
  - každou iterací je obsah znovu vytvořen
  - současný přístup v režimu read-only na vícejádrovém procesoru
  - po aplikaci operátorů mohou vznikat stejné nové morfy, ale následnou filtrací (bod 5 pseudokódu) se do stromu dostane jen unikát

V předchozí sekci o datových strukturách byla nastíněna myšlenka rozdělení výpočtu na cluster multiprocesorů. Důvodem proč přechod na cluster má cenu jsou body 2, 3, 4 a 5 pseudokódu, které představují dostatečně velkou porci rozdělitelné práce na všechny uzly clusteru s malou režií (také ale záleží na parametrech

algoritmu). Oproti tomu body 1, 2-a, 2-b a 6 využívají relativně malé sdílené datové struktury a proto nejsou vhodné pro všechny uzly clusteru, ale raději pro jeden vícejádrový procesor.

I z toho, co bylo napsáno, je očividné, že obě úrovně paralelizace mohou být vzájemně kombinované. Každopádně úroveň clusteru je větší výzva i z důvodu většího zrychlení. Má smysl rozdělit paralelizaci do více kroků podle jejich náročnosti a výhod. V prvním kroku byl Molpher paralelizován pro vícejádrové procesory, což samo o sobě poměrně dost urychlilo výpočet. V budoucím případném dalším rozvoji by stálo za to rozšířit stávající paralelizaci pro počítačové clustery (i když se zrychlí jen některé části programu).

1.5 Výstup

Molpher na konci výpočtu, po nalezení cesty, vydá tři různé výstupní soubory:

- **path.txt** obsahuje textovou reprezentaci molekul (ve tvaru SMILES) na nalezené cestě, které mohou být lehce rozparsované. Na prvním řádku je zdrojová molekula, na druhém informace o použitém morfovacím operátoru ve tvaru *id_operatoru:zkratka_operator*, na třetím vzniklá molekula po použití onoho operátoru a tak dále až na posledním řádku je cílová molekula. Takže molekuly se vyskytují na lichých řádcích a operátory na sudých.

- **final_mols.sdf** obsahuje kandidátské molekuly z poslední iterace. Nejen tedy molekuly na cestě, ale i ostatní kandidáty, kteří nestačili být prořezáni.

- **all_mols.sdf** je pravděpodobně o něco užitečnější soubor molekul než ten předešlý. Obsahuje všechny kandidátské molekuly, které za celou dobu výpočtu existovaly. Je to tedy sjednocení kandidátských stromů všech iterací výpočtu.

Nejpřínosnější je soubor *all_mols.sdf*, který slouží jako vhodná obsáhlá množina jak výsledných molekul na cestě, tak okolí cesty. Tento soubor je tedy vhodný pro další testování nejdříve algoritmicky a poté i reálně v laboratořích.

1.6 Ukázka použití

Podrobnější informace o celém projektu Molpher včetně programátorské i uživatelské příručky jsou k dispozici v online podobě [18]. Zde jen ukáži základní práci s Molpherem, která spočívá v připojení se k serveru a zadání úlohy hledání cesty (tzv. jobu):

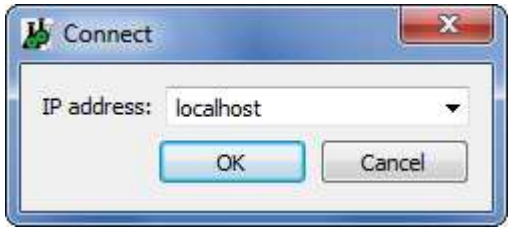
- Ujistěte se, že víte kde je server spuštěn – klient samotný může prohlížet již vytvořené stavy výpočtu uložené v SNP souborech, ale nyní chceme zadat nový výpočet.

- Spusťte klienta.

- Připojte klienta k serveru použitím tlačítka "Connect" (viz obrázek 1.6). V případě, že se obě části nacházejí na stejném počítači, je klient již připojen.

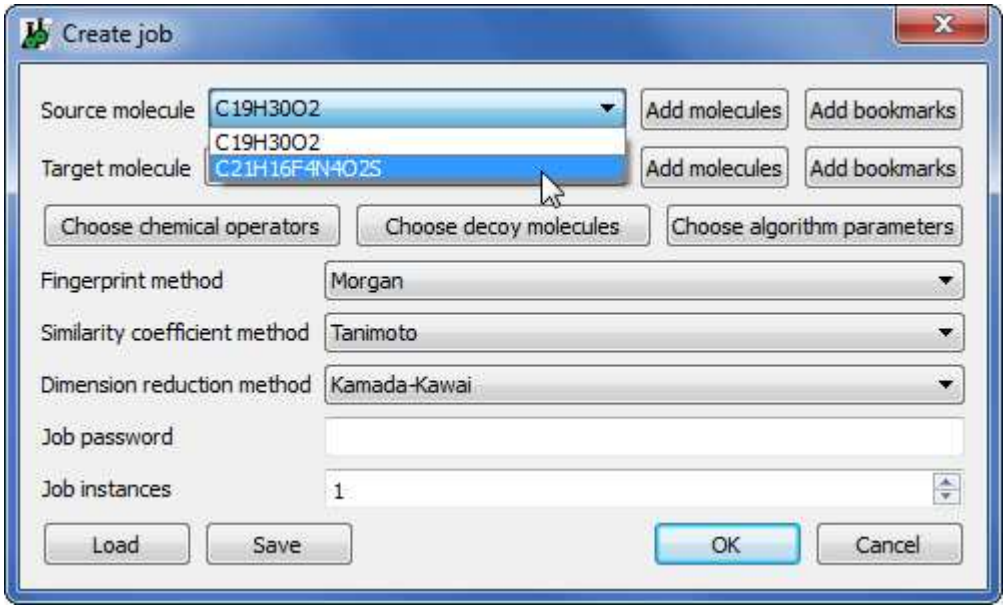
Obrázek 1.6: Odpojený klient.

- Jsou zde dvě možnosti (obrázek 1.7):
  - Připojení na server, který se nachází na stejném stroji. V tom případě stačí napsat "localhost" a potvrdit dialog.
  - Vzdálené připojení – vybere se IP adresa z historie skryté v combo boxu nebo se zadá nová a potvrdí se dialog.

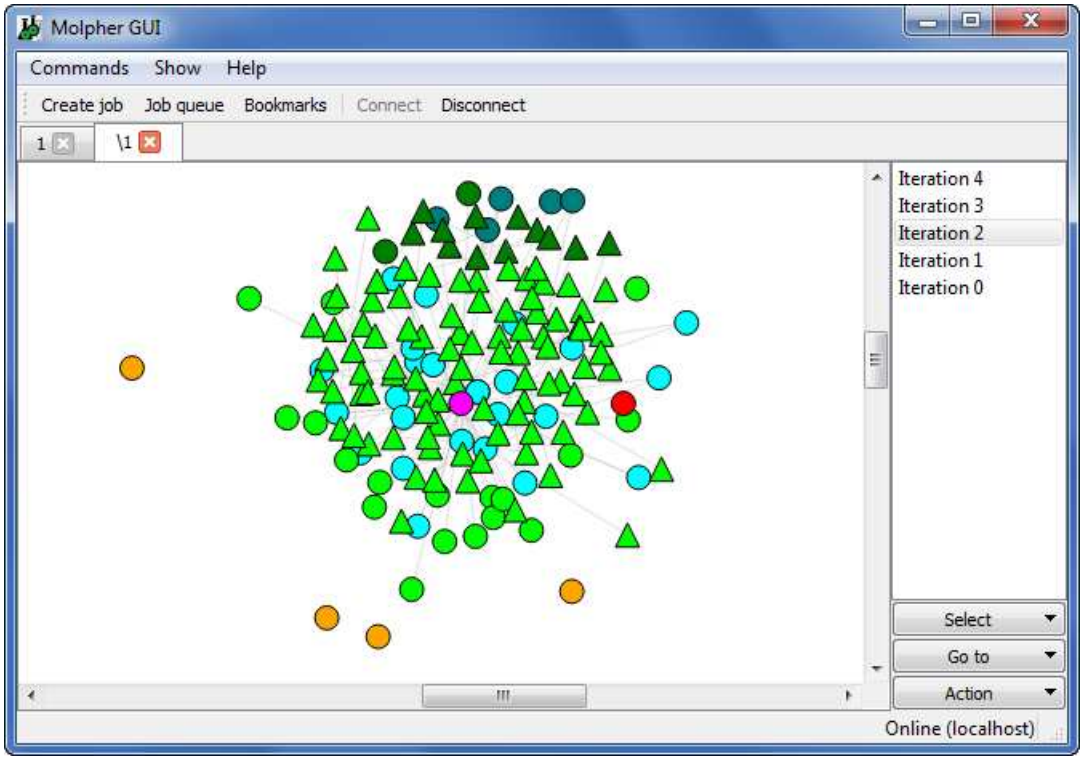


Obrázek 1.7: Dialog připojení.

- Klikněte na tlačítko "Create job", objeví se nový dialog (obrázek 1.8).
- Existují dvě možnosti nastavení dialogu:
  - Načtení všech dat z předem vytvořeného snapshotu tlačítkem "Load".
  - Manuální vyplnění.
    - Zdrojová a cílová molekula se načítá buď z SDF souboru pomocí tlačítka "Add molecules" nebo záložek do kterých v minulosti uživatel vložil pro něho zajímavou molekulu ("Add bookmarks").
    - Ostatní položky jsou defaultně (první spuštění) nastavené nebo tak, jak je uživatel nastavil při posledním otevření dialogu. Jejich významy jsou opět popsané v uživatelské příručce [18].
- Po potvrzení dialogu server začne počítat danou cestu. Uživatel může sledovat stav jobu v části "Job queue", kde se po kliknutí na tlačítko "Live" zobrazí grafické znázornění stavu výpočtu a stále se vykresluje aktuální stav. Pro prohlížení stavů výpočtu do nejnovějšího slouží tlačítko "Detach", kde je tab označen zpětným lomítkem a ID jobu. Toho bylo využito i na obrázku 1.9, kde se zobrazuje stav ve druhé iteraci.



Obrázek 1.8: Dialog vytvoření jobu.



Obrázek 1.9: Prohlížení stavů výpočtu.

2. Scaffold hopping

Motivací vypsání této magisterské práce byla idea, že nalezení cesty mezi danou zdrojovou a cílovou molekulou je znatelně rychlejší, když je známa prostřední molekula z cesty a hledají se dvě poloviční cesty, než když se celá cesta hledá najednou. Je tedy lepší spustit program na více kratších cest než na jednu dlouhou. K tomu by měla technika scaffold hopping pomoci.

Dále v textu diplomové se bude používat pojem *granularita* a proto tento odstavec slouží pro jeho objasnění. Granularita je definována jako úroveň rozlišení molekul. Dalo by se to přirovnat k tomu, kdy dobře vidící člověk použije dioptrické brýle s různými hodnotami dioptrií. Čím silnější brýle si vezme, tím hůře bude vidět a tím méně detailů rozpozná. Stejné je to se scaffoldy. Různé úrovně se na molekuly dívají s různým rozostřením a tedy různou přesností.

Scaffold hopping je technika, která objevuje strukturně nové molekuly takovým způsobem, že se molekule upravuje její kostra – například se odebere nebo přidá kruh. Principem scaffold hoppingu je tedy nalézt nové chemické sloučeniny s podobnou biologickou aktivitou jako má původní sloučenina a to tak, že se nahradí některé její části. Částmi sloučeniny mám na mysli větší části jako již zmíněné kruhy a ne jednotlivé atomy či vazby. Tohoto lze dosáhnout při nahlížení na sloučeniny s různou úrovní granularity a následným aplikováním morfovacích operátorů. Díky změnám v kostře se chemický prostor neprochází po krocích, ale prochází se po "skocích" – skákat se anglicky řekne *hop* a proto se používá termín scaffold hopping.

Předpokládá se, že výsledné molekuly budou strukturně rozmanitější kvůli "skákání" prostorem chemických molekul, což je výhodné v drug discovery procesu [7, 35]. Drug discovery proces je proces hledání léčiv, kde je využíváno mnoho přístupů a dá se říci, že všechny jsou časově i finančně náročné. Nalézáním nových molekul s odlišnými strukturami získáme rozmanitější množinu potenciálně užitečných sloučenin a tím se zvyšuje pravděpodobnost nalezení léčiv. V drug discovery je tato strukturní rozmanitost ceněnou vlastností a bude zkoumána v kapitole věnované testům.

Nyní se dostávám k samotné modifikaci aplikace, kterou jsem provedl v rámci této diplomové práce. Cílem bylo navrhnout a implementovat algoritmus, který bude umožňovat procházení prostoru chemických sloučenin pomocí techniky scaffold hopping, kdy je molekula reprezentována zjednodušenou formou (tzv. scaffoldem).

Pro správné fungování techniky scaffold hopping bylo potřeba definovat několik úrovní scaffoldů díky jimž se hledaná cesta prozkoumává postupně od nejobecnější úrovně. Na každé úrovni se objeví část cesty a po nejnižší úrovni bude nalezena celá cesta. Složitost následného procházení chemického prostoru závisí na granularitě každé úrovně. S tím souvisí i množina operátorů, která se bude pro každou úroveň lišit. Abych to přiblížil, například pro scaffold úrovně, které ignorují a neberou v úvahu takzvané "slepé cesty" (atomy a vazby, které nejsou v kruzích ani na cestách mezi nimi) nemá cenu používat morfovací operátor, který přidává jeden atom. Tento přidaný jeden atom, s vazbou na molekulu, je vlastně také slepá cesta, změna vede ve stejný scaffold a tudíž na dané úrovni granularity nemá smysl.

Každá úroveň granularity může být různými způsoby definovaná. Je ale logické zvolit úrovně tak, aby se je dalo jasně seřadit od nejobecnější po nejkonkrétnější – aby obecnější úroveň byla podmnožinou nižší úrovně. Jak jsem napsal, úroveň původních molekul je také scaffold úroveň – ta poslední a nejkonkrétnější úroveň granularity. Tudíž jsem jejich pořadí, v jakém se budou provádět, vybral tak, aby nebyly zbytečné. Nejvyšší úroveň scaffoldů se provádí jako první a nejnižší úroveň jako poslední. Kdyby teoreticky úroveň původních molekul byla nejvyšší (a ne nejnižší, jak tomu je), našla by se celá cesta hned na této úrovni a zbylé by byly bezvýznamné. Neměly by smysl, protože by po následném spuštení hledání cesty mezi dvěmi molekulami hned skončily.

Počáteční myšlenka fungování Molpheru s technikou scaffold hopping byla taková, že se v nejobecnější (nejvyšší) úrovni granularity nalezne cesta. Tato cesta bude obsahovat libovolný počet nových molekul mezi zdrojovou a cílovou molekulou. Tyto molekuly budou součástí případné celkové cesty (pokud se cesta nalezne) po skončení celého výpočtu. V další konkrétnější (nižší) úrovni se pro každou dvojici molekul na celkové cestě znovu hledá cesta a nové molekuly jsou zařazeny do výsledné cesty mezi danou dvojici molekul. Tedy v případě nalezení jedné nové molekuly na nejvyšší úrovni se na nižší úrovni budou hledat dvě cesty – mezi zdrojem a novou molekulou z nejvyšší úrovně a mezi novou molekulou z nejvyšší úrovně a cílem. Tento postup se opakuje a končí se na nejnižší úrovni granularity, což jsou původní molekuly. Tedy abych použil mé přirovnání z prvního odstavce, nakonec se cesty hledaní s nasazenými brýlemi bez sklíček. Takto nahradíme hledání jedné dlouhé cesty hledáním více krátkých cest. Krátkých, a tedy rychle nalezených cest, bude mnoho, navíc vytváření a modifikace scaffoldů bude stát nějaký čas. Z tohoto důvodu je tedy otázka, zda se nakonec celková cesta nalezne časově rychleji či nikoliv. Celkový počet iterací by měl být nižší.

Následující sekce bude pojednávat o samotném výběru scaffoldů a jejich popisu.

2.1 Definice scaffoldů

Při definici scaffoldů jsem se inspiroval programem Strip-it [34]. V něm se generují různé druhy, tudíž to byla ideální volba. Jedná se o konzolovou aplikaci generující scaffoldy, vstupy jsou molekula a úroveň scaffoldu a výstupen je scaffold tvar vstupní molekuly. Pro práci s molekulami využívá C++ knihovnu OpenBabel.

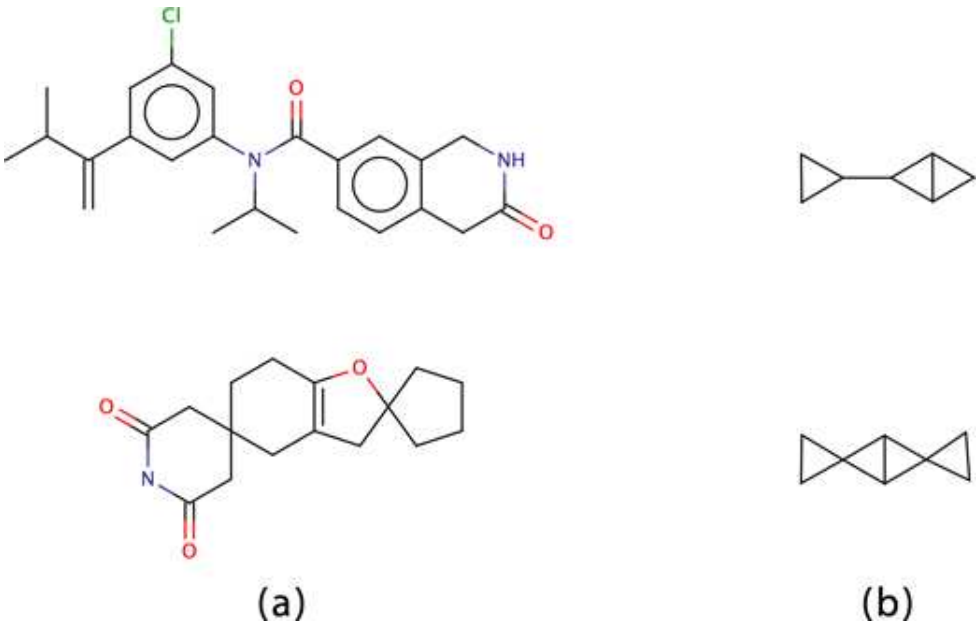
Tento program disponuje čtyřmi základními typy scaffoldů – *Rings with linkers*, *Murcko*, *Oprea* a *Schuffenhauer*. Každý se dále ještě dělí. Jednotlivé podtypy si jsou většinou velmi podobné. Po zvážení situace bylo rozhodnuto, že celkově ctyři úrovně scaffoldů budou dostatečné. Více úrovní by mohlo znamenat lepší výsledky pro dlouhé cesty mezi zadanými molekulami, nicméně pro tuto první verzi je to dostatečné.

Byly vybrány následující úrovně scaffoldů v pořadí od nejobecnější: *Oprea 1*, *Murcko 2* a *Rings with linkers 1.* A jako čtvrtá úroveň jsou původní molekuly. Dále budou jednotlivé typy popsány.

2.1.1 Oprea 1

Scaffold vznikne následujícím způsobem (*stupeň atomu* je definován jako počet různých atomů, ke kterým vede vazba; například atom má stupeň jedna, když z něj vede jedna dvojitá vazba):

- odstraní se všechny atomy stupně jedna
- zbylé atomy se změní na neutrální uhlík
- všem vazbám se nastaví vazebný řád jedna (jednoduchá vazba)
- cesty spojující kruhy se zmenší na velikost jedna
- z kruhů jsou vynechány atomy, které mají vazby pouze v rámci kruhu a jejimž vynecháním bude kruh stále existovat (musí mít velikost alespoň tři)



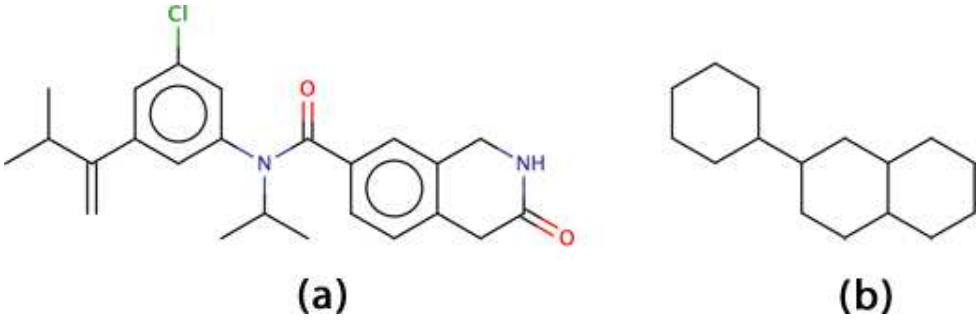
Obrázek 2.1: (a) Původní molekuly, (b) Oprea 1 scaffoldy.

Tento typ je velmi podobný typu *Murcko 2* s tím rozdílem, že pro *Murcko 2* se velikosti kruhů nemění (poslední bod modifikace je vynechán).

2.1.2 Murcko 2

Tento typ vznikne následující úpravou:

- odstraní se všechny atomy stupně jedna
- zbylé atomy se změní na neutrální uhlík
- všem vazbám se nastaví vazebný řád jedna (jednoduchá vazba)
- cesty spojující kruhy se zmenší na velikost jedna

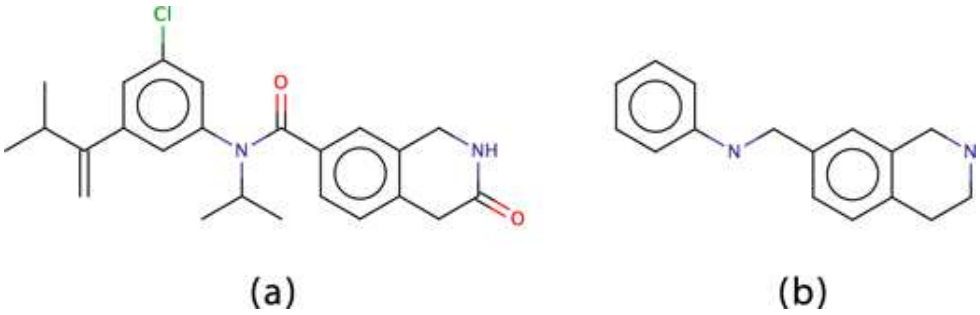


Obrázek 2.2: (a) Původní molekula, (b) Murcko 2 scaffold.

2.1.3 Rings with linkers 1

Tento typ je nejvíce podobný původní molekule, proto je použit při výpočtu jako třetí před úrovní původních molekul. Pro jeho vytvoření je potřeba následující úprava:

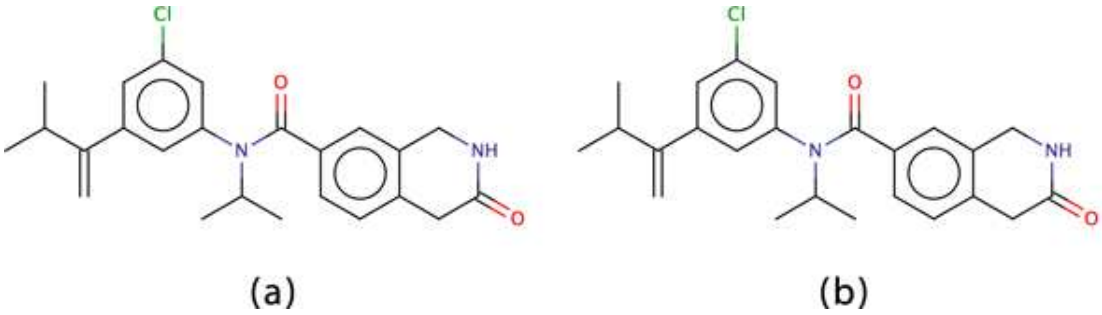
- odstraní se všechny atomy stupně jedna



Obrázek 2.3: (a) Původní molekula, (b) Rings with linkers 1 scaffold.

2.1.4 Původní molekula

Zde není mnoho, co popisovat. Jedná se o regulérní úroveň scaffoldu a na této úrovni se tedy molekula nemodifikuje.



Obrázek 2.4: (a) Původní molekula, (b) Scaffold úrovně původních molekul (beze změn).

2.2 Implementace

Použití morfovacích operátorů

Nyní se dostanu k tomu, jak konkrétně probíhá samotné generování nových molekul (morfů) stromu kandidátů.

Původní plán byl takový, že by se molekuly upravovaly na úrovních jednotlivých scaffoldů. Při hledání cesty by se používaly morfovací operátory (vhodné pro aktuální úroveň) na daný scaffold, ne na jeho reálnou podobu. Pracovalo by se vždy na úrovni scaffoldů. Nedocházelo by k tomu, že by vznikl stejný scaffold – Molpher nepoužívá operátory, které nic nedělají. Po nalezení cesty na nějaké úrovni granularity by se scaffoldy z výsledné cesty musely převést na reálné molekuly, minimálně na vzhled scaffoldu pro další (konkrétnější) úroveň. A toto je problém. Jak by na konkrétnější úrovni *Murcko 2* měly vypadat kruhy o velikosti tři i více z úrovně *Oprea 1*? Jak transformovat nově nalezenou molekulu z cesty na konkrétnější scaffold úroveň? Kruhy mohou vypadat různě a nelze přesně říci jakými atomy či vazbami by byly spojeny. Možností vzhledů kruhů už na *Oprea 1* úrovni je mnoho a nelze uvažovat o tom, že by tyto informace dodal uživatel. To platí i pro úroveň *Murcko 2*. Proto byla vybrána strategie, kde se používají operátory jen a pouze na úrovni původních molekul a na sloučeniny se nahlíží z úrovně aktuálního scaffoldu.

Je to na první pohled komplikovanější, protože se vždy pracuje jak na scaffold úrovni, tak na úrovni samotných molekul. Nicméně to je nutné a při zdůraznění tohoto faktu snad i lehce pochopitelné. Funguje to tak, že se vždy jednotlivé operátory používají na původní molekuly (tedy na nejnižší úroveň scaffoldů), ale aplikace se na ně dívá s různou úrovní granularity. Pro každou úroveň se používají smysluplné operátory (viz sekce 2.2.1) na reálnou molekulu, na níž je pak nahlíženo jako na daný scaffold. Operátory jsou vybírány tak, aby ovlivnily výsledný vzhled.

Může se stát, že použitím nějakého operátoru se samotná molekula změní, ale scaffold ne. V tom případě je o scaffoldu rozhodnuto, že nedošlo ke zlepšení (nezměnil se) a vytvořený morf je zahozen. Způsob používání morfovacích operátorů může být hůře pochopitelný kvůli pracování na dvou úrovních současně (jak daná úroveň granularity, tak samotné molekuly), ale veliká výhoda spočívá v tom, že v každé chvíli výpočtu je známa reálná podoba molekuly.

Hledání podcest

Bylo tedy rozhodnuto o tom, že se morfovací operátory budou používat na původní molekuly a dále se bude pracovat s jejich scaffoldy. Pokud tedy má například molekula kruh o šesti vazbách, její scaffold úroveň *Oprea 1* zobrazuje kruh zmenšeně se třemi vazbami, pak morfovací operátor Remove Bond odebere jednu ze šesti vazeb skutečné molekuly, nikoliv jednu ze tří příslušného scaffoldu.

Plánovalo se, že na nejvyšší úrovni scaffoldů se nalezne cesta pro ilustraci s jednou novou molekulou. Takže po nejvyšší úrovni by prozatimní cesta měla tři molekuly – zdrojovou, nově nalezenou a cílovou molekulu. Na nižší úrovni se nejdříve bude hledat cesta mezi zdrojem a nově nalezenou molekulou z nejvyšší úrovně a pak mezi nově nalezenou a cílovou molekulou. Z obou cest na vyšší úrovni se vygenerují další hledané molekuly výsledné cesty. Následně se přejde

do další nižší úrovně a postup se opakuje (hledají se cesty postupně mezi sousedními sloučeninami z cesty). Jelikož se ale změnil styl používání morfovacích operátorů, ve smyslu že se používají na původní molekuly, i výše zmíněné generování cest bylo potřeba upravit. To proto, že prvních $n-1$ molekul z aktuálního stavu cesty o délce n se měnit nebude (tyto molekuly nakonec budou na výsledné cestě). Vznikly použitím jednoho morfovacího operátoru na předešlou molekulu, takže mezi nimi se již cesta hledat nemusí. Ve světle těchto skutečností se tedy vždy bude na každé další úrovni scaffoldů generovat cesta jen mezi předposlední molekulou z cesty a cílovou. Jen tento prostor je potřeba prozkoumat, protože přechodem na konkrétnější úroveň se tyto molekuly "vzdálily".

2.2.1 Výběr morfovacích operátorů

Morfovací operátory pro první tři nejvyšší úrovně scaffoldů jsou pevně dané, uživatel je nemůže nastavit. Kdyby mohl, hrozilo by velké riziko toho, že se scaffold nebude moci vyvíjet a měnit svůj tvar. Pro úroveň *Oprea 1* nemají čtyři morfovací operátory z osmi smysl. Při dalším omezení této množiny by tedy mohl nastat problém v tom smyslu, že by se nenašla cesta.

Jen čtvrtá nejnižší úroveň tvoří výjimku. Jedná se totiž o všem chemikům dobře známou úroveň původních molekul, kde Molpher dává uživateli volnost. Jelikož má tedy uživatel při klasickém hledání cesty možnost volby operátorů, má tuto možnost i při zapnutém scaffold hoppingu na poslední úrovni.

V tabulkách s čísly 2.1, 2.2 a 2.3 jsou informace o operátorech, které se pro danou úroveň používají, a případné poznámky omezující množinu atomů/vazeb se kterou daný operátor pracuje.

Oprea 1

Pro typ scaffoldu *Oprea 1* je zakázán morfovací operátor *Add Atom* z toho důvodu, že nezmění scaffold molekuly. Jinak řečeno reálné molekule se přidá nový atom, ale z pohledu *Oprea 1* se nic nestane, jelikož atomy stupně 1 jakoby nevidí. Tudíž v žádném případě na této úrovni nemá daný operátor smysl. Ze stejného důvodu (Molpher maže pouze atomy stupně jedna) nelze použít operátor *Remove Atom*.

V případě morfovacího operátoru *Interlay Atom* není jeho zakázání tak očividné. Způsobí rozdvojení vazby, tudíž zvětšení například cesty mezi kruhy nebo samotného kruhu atd. *Oprea 1* úroveň scaffoldu ale zmenšuje kruhy i cesty na minimum a zbylé cesty úplně ignoruje. Takže toto rozdvojení vazby by opět nezměnilo scaffold.

Proč nemá cenu používat morfovací operátor *Mutate Atom* je zřejmé – na této úrovni granularity se na všechny atomy nahlíží jako na neutrální uhlíky a změna atomu na úrovni původních molekul opět nezmění scaffold.

Zbylé operátory mají alespoň v některých případech smysl. *Add Bond* má vždy smysl. Vždy vytvoří kruh, protože mezi danými dvěmi atomy již před tím existovala cesta (jinak by atomy nebyly ze stejné molekuly).

Dále *Bond Contraction* má v některých případech smysl a to když vazba je obsažena v kruhu, kde krajní atomy vazby mají stupeň alespoň tři (dvě vazby jsou součástí kruhu a alespoň jedna vede mimo kruh), nebo přímo spojuje dva

kruhy. V případě, kdy přímo spojuje dva kruhy, se provedením operátoru tyto kruhy spojí a budou mít společný jeden atom namísto vazby. To se projeví i na scaffoldu tak, že minimální kruhy se budou dotýkat jedním atomem (před tím mezi nimi byla ona vazba). Může se ale stát i to, že použití operátoru nezmění scaffold. To se stane v případě, kdy je vazba obsažená v kruhu a její krajní atomy mají stupeň dva (z krajních atomů nevede vazba mimo kruh). Již před modifikací operátorem *Bond Contraction* byla při transformaci na *Oprea 1* scaffold tato vazba odstraněna a kruh zmenšen na minimální velikost, takže tato modifikace se neprojeví při následné transformaci na *Oprea 1* scaffold a zůstane stejný. Nezmění se a tudíž se tento morf následně zahodí.

Třetí použitelný operátor je *Bond Reroute*, který je užitečný jen pro vazby v kruzích nebo na cestách mezi kruhy. Opět platí to, že ne vždy výsledný scaffold změní svojí podobu. To se může stát například v případě, kdy dva kruhy jsou spojené cestou o více vazbách a prostřední se přepojí ke kruhu tak, že cesta mezi kruhy se zmenšila. Jelikož cesty mezi kruhy na úrovni *Oprea 1* mají délku jedna, scaffold se nezmění.

A jako poslední použitelný operátor je *Remove Bond*, který je bez omezení oproti původní implementaci. Je to způsobené tím, že od své podstaty se nemůže používat na slepých cestách – jinak by se molekula rozpadla. Způsobuje rozpadnutí kruhu.

Operátor	Použit	Omezení
Add Atom	Ne	
Add Bond	Ano	
Bond Contraction	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy o velikosti 1
Bond Reroute	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy - ne na slepých cestách
Interlay Atom	Ne	
Mutate Atom	Ne	
Remove Atom	Ne	
Remove Bond	Ano	

Tabulka 2.1: Oprea 1 – morfovací operátory.

Murcko 2

Množina požitých operátorů pro úroveň *Murcko 2* se zvětšila o *Interlay Atom*. Jelikož v této úrovni se již velikosti kruhů při transformaci na scaffold nemění, vložením nového atomu do kruhu na úrovni původních molekul se zvětší kruh jak samotné molekuly, tak i scaffoldu. Důvody, proč zbylé operátory jsou použity či nikoliv, byly popsány v předešlé části věnující se úrovni *Oprea 1*.

Rings with Linkers 1

Rings with Linkers 1 ponechává typy atomů a netransformuje je na neutrální uhlík, tudíž má smysl použít chemický operátor *Mutate Atom* na vazby v kruzích nebo na cestách mezi kruhy.

Operátor	Použit	Omezení
Add Atom	Ne	
Add Bond	Ano	
Bond Contraction	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy o velikosti 1
Bond Reroute	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy - ne na slepých cestách
Interlay Atom	Ano	Jen pro vazby v kruzích
Mutate Atom	Ne	
Remove Atom	Ne	
Remove Bond	Ano	

Tabulka 2.2: Murcko 2 – morfovací operátory.

Dále jde o stejnou logiku jako u *Oprea 1*, kde jsou důvody pečlivě popsané.

Operátor	Použit	Omezení
Add Atom	Ne	
Add Bond	Ano	
Bond Contraction	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy - ne na slepých cestách
Bond Reroute	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy - ne na slepých cestách
Interlay Atom	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy - ne na slepých cestách
Mutate Atom	Ano	Jen pro vazby v kruzích nebo na cestách mezi kruhy - ne na slepých cestách
Remove Atom	Ne	
Remove Bond	Ano	

Tabulka 2.3: Rings with Linkers 1 – morfovací operátory.

Původní molekula

Jak již bylo napsáno, pro tuto poslední úroveň si vybírá uživatel množinu morfovacích operátorů sám. Žádná omezení zde nejsou.

2.2.2 Generování scaffoldů

Jelikož knihovna RDKit neobsahovala potřebnou fukcionalitu, která by převedla vstupní molekulu na scaffold, byl jsem donucen tyto třídy naimplementovat. Stejně jako jsem se nechal inspirovat programem Strip-it [34] (používá knihovnu Open Babel) při definici úrovní scaffoldů, tak jsem ho měl jako vzor i při implementaci. Posloužil jako dobrý návod při vytváření oněch tříd, které generují scaffoldy. Byly přepsány tři druhy – *Oprea 1*, což je nejobecnější představitel, dále *Murcko 2* a *Rings with linkers 1*. Posledně zmiňovaný scaffold je z těchto tří nejblíže původní molekule, nicméně neni nejkonkrétnějším scaffoldem, protože tím

je samotná původní molekula. Při hledání cesty se tedy procházejí čtyři úrovně – tři výše zmíněné a jako poslední úroveň samotných molekul.

Pro mé účely stačily výše zmíněné tři úrovně scaffoldů, nicméně není problém se dále inspirovat knihovnou Open Babel a přidat do Molpheru další.

Bázová třída, ze které jsou všechny typy scaffoldů odvozeny je následující (je vidět pouze stěžejní rozhraní třídy):

```
class Scaffold
{
public:
    void GetScaffold(std::string &mol, std::string *scaff);
    // caller is responsible for deallocation
    virtual void GetScaffold(std::string &mol, RDKit::RWMol **scaff) = 0;
    virtual ScaffoldSelector GetSelector() = 0;
    virtual std::vector<ChemOperSelector> GetUsefulOperators() = 0;

protected:
    ...
};
```

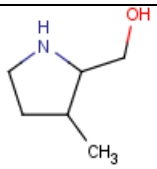
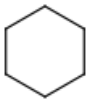
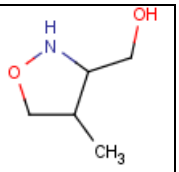
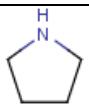
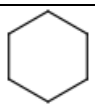
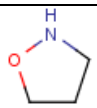
Rozhraní se skládá ze čtyř metod – dvě vracejí výslednou podobu scaffoldu buď jako SMILES typu string nebo jako RDKit molekulu typu *RDKit::RWMol*, další (*GetSelector()*) vrací idetifikátor daného scaffoldu a poslední (*GetUsefulOperators()*) seznam použitelných morfovacích operátorů. Metoda vracející výslednou podobu scaffoldu jako SMILES není virtuální, protože je vždy stejná. Získává RDKit molekulu zavoláním virtuální funkce *GetScaffold* a následně vrátí scaffold SMILES dané molekuly.

Nakonec popíši situaci, která nastávala nově při technice scaffold hopping, ale nakonec nebyl problém se s ní vypořádat. Při vypnuté technice scaffold hopping všechny morfy obsahují atomy, které se vyskytují buď ve zdrojové nebo cílové molekule. Přesněji řečeno, morfy vygenerované z listu obsahují atomy z daného listu nebo celkového cíle. Jiné atomy nemá cenu brát v úvahu – morf by se touto modifikací nepřiblížil cíli, ale naopak. Situace, kdy morf může obsahovat nový atom oproti listu ze kterého vznikl, se týká morfovacích operátorů *Add Atom*, *Interlay Atom* a *Mutate Atom*. Jak již bylo řečeno a důvody obhájeny, při technice scaffold hopping se modifikují molekuly na úrovni původních molekul a následně je na ně nahlíženo z pohledu scaffoldů. To může způsobit situaci, že scaffold morfu může obsahovat atom, který se nevyskytuje ve scaffold listu ani ve scaffold tvaru cílové molekuly (viz tabulka 2.4 – výsledný morf je vidět na posledním řádku a posledním sloupci).

2.2.3 Modifikace Molpheru

Co se týče úpravy původní verze Molpheru, velice důležité bylo rozmyslet a následně upravit stávající datové struktury představující stav výpočtu (*IterationSnapshot*) a informace o molekule (*MolpherMolecule*). Stejně tak ale i to, jakým způsobem bude funkcionalita scaffold hopping implementována. Zda pomocí zcela nových objektů nebo přidáním ke stávající infrastruktuře.

Součástí mé modifikace byla i změna výstupu Molpheru – tedy výstupních souborů. Soubor *path.txt* se upravil tak, že na stejném řádku, kde se nachází

	List	Cíl	Morf
Reálná molekula			
RWL1 scaffold			

Tabulka 2.4: Nová situace vzhledu morfu ukázaná na scaffold úrovni Rings with linkers 1.

molekula z cesty, je i informace o scaffold úrovni ve které byla vytvořena. Na prvním řádku se zdrojovou molekulou je informace o tom, že nebyla na žádné úrovni vytvořena, protože byla zadána (a z ní se generovaly morfy). Na posledním řádku s cílovou molekulou je uvedena nejkonkrétnější úroveň – sice byla tato molekula také zadána, ale hlavně byla vytvořena jako list stromu. Při zapnuté technice scaffold hopping přestal být zajímavý soubor *final_mols.sdf* (obsahuje kandidátské molekuly z poslední iterace) a proto se vůbec negeneruje. Bez dané techniky se generuje pouze jedna cesta a molekuly ze souboru mohou existovat již od počátku výpočtu. Se zapnutou technikou by tento soubor obsahoval přeživší molekuly z hledání poslední cesty – tedy ne od úplného začátku generování cesty. Třetí a poslední soubor *all_mols.sdf* je stále stejně důležitý a zůstal beze změny.

Struktuře představující molekulu (*MolpherMolecule*) bylo potřeba přiřadit informaci o scaffold úrovni, kdy vznikla, a samotný SMILES daného scaffoldu. Úroveň je mimo jiné potřeba po skončení hledání cesty, kdy se ke každé molekule na cestě v souboru *path.txt* přidává informace o scaffold úrovni vytvoření. SMILES scaffoldu je důležitý pro porovnávání při výpočtu a v budoucnu může být použit i v klientské části pro zobrazení. Bylo rozhodnuto o tom, že nevznikne potomek struktury *MolpherMolecule*, který by obsahoval zmíněné dvě položky a zbytek informací by byl v předku. Důvod je ten, že se jedná o strukturu s veřejnými proměnnými a ne o třídu s virtuálními metodami, kde by se využil polymorfismus. Proto byly do již existující struktury přidány nasledující dvě položky.

```
struct MolpherMolecule
{
    ...
    std::string scaffoldSmile;
    boost::int32_t scaffoldLevelCreation;
    ...
}
```

Struktura *IterationSnapshot* představuje stav výpočtu. Uživatel může vytvořit nový job tak, že načte libovolný SNP soubor znamenající libovolnou iteraci při hledání nějaké cesty a tím začne výpočet z daného stavu. Proto je potřeba do této struktury přidat další informaci o dočasném zdroji, dočasném cíli, aktuální scaffold úrovni nebo již vypočítaných molekulách na cestě. Dále nejen kvůli rychlému vyhledávání (logaritmická složitost) jsou ve struktuře dvě mapy. Jedna pro

vyhledávání mezi molekulami na cestě (*pathScaffoldMolecules*) a druhá pro vyhledávání mezi kandidáty (*candidateScaffoldMolecules*), kterých může být stovky či tisíce. Ze stejného důvodu jako v případě struktury *MolpherMolecule* i zde jsou tyto položky přímo ve struktuře a nevytvářel se její potomek. Následuje výřez těchto nových položek ve struktuře představující jeden SNP soubor:

```
struct IterationSnapshot
{
    ...
    typedef std::map<std::string, std::string> ScaffoldSmileMap;
    ...
    /// variables used for scaffold hopping
    MolpherMolecule tempSource;
    MolpherMolecule tempTarget;
    boost::int32_t scaffoldSelector;
    std::vector<MolpherMolecule> pathMolecules;
    ScaffoldSmileMap pathScaffoldMolecules;
    ScaffoldSmileMap candidateScaffoldMolecules;
};
```

Přidáním těchto položek samozřejmě vzrosta velikost SNP souboru. Nejvíce je to znatelné pro iniciální snapshoty, protože je ještě prázdná mapa kandidátů. Záleží na velikosti SMILES řetězců zadaných molekul. Nejvíce místa zabírají struktury *MolpherMolecule*, které jsou nejen zdrojovou a cílovou molekulou, ale i dočasným zdrojem a cílem a dvěmi molekulami na cestě. S přibývajícími iteracemi, kdy stoupá počet kandidátských molekul, se však tento rozdíl stává zanedbatelný.

Výpočetním jádrem celé serverové části je třída *PathFinder*, která provádí hledání cesty mezi zdrojovou a cílovou molekulou. Tato třída používá strukturu *PathFinderContext* jako interní stav výpočtu. V podstatě se jedná o obraz *IterationSnapshot*, který ovšem používá struktury podporující konkurenční přístup při paralelním používání. Jednotlivé části byly upraveny takovým způsobem, aby zvládaly novou techniku scaffold hopping. Muselo se dát pozor na to, aby se vždy pracovalo se správnou molekulou (původní molekulou nebo scaffoldem). Při počítání vzdálenosti k cíli se provádí vůči scaffoldu cíle, nikoliv jeho reálné podobě. Během redukce dimenze se nesmí zapomenout i na již nalezené molekuly na cestě.

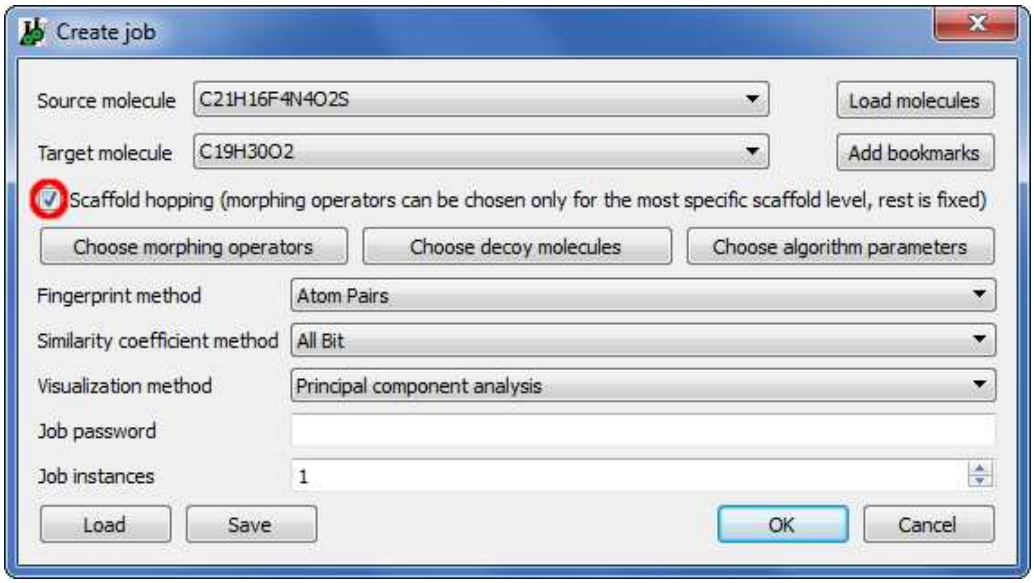
Jednou z největších úprav na serverové části programu bylo ošetření konce iterace, kdy se kontroluje, zda byla nalezena cesta (jestli se má generovat další) a případně se mění stav výpočtu.

Důraz byl kladen na to, aby se scaffold molekuly vytvářely co možná nejméně. Každá práce s molekulami stojí čas. Proto se používá mapa molekul na cestě a mapa kandidátských molekul. Kdyby nebyla mapa molekul z výsledné cesty zadefinovaná, muselo by se pracovat jen s již existujícím vektorem těchto molekul. Každá molekula z tohoto vektoru obsahuje informaci o scaffold úrovni, kdy byla vytvořená, a příslušný SMILES scaffoldu. Tyto informace musí zůstat neměnné pro zjištění, kdy jaká molekula byla vytvořena. Takže při absenci mapy molekul z výsledné cesty by se v každé iteraci musel zjišťovat jejich scaffold a zbytečně by se výpočet zpomaloval.

Pro mapu kandidátů neplatí tvrzení, že by se v každé iteraci musel scaffold SMILES zjišťovat, protože tato informace je již obsažená v každém kandidátu (byl vytvořen na aktuální scaffold úrovni a tudíž jeho scaffold SMILES je použitelný). Nicméně zde je důležitý jiný fakt a to, že vyhledávání v mapě má logaritmickou složitost, což v případě množiny o stovkách i tisících kandidátů je vhodné.

2.3 Ukázka použití

Nyní ukáži postup, kterým se vygeneruje cesta v chemickém prostoru pomocí techniky scaffold hopping. Bude se začínat v dialogu vytvoření jobu. Předchozí spuštění a připojení se je vynecháno, protože již bylo popsáno v kapitole 1.6 věnující se původnímu projektu Molpher.



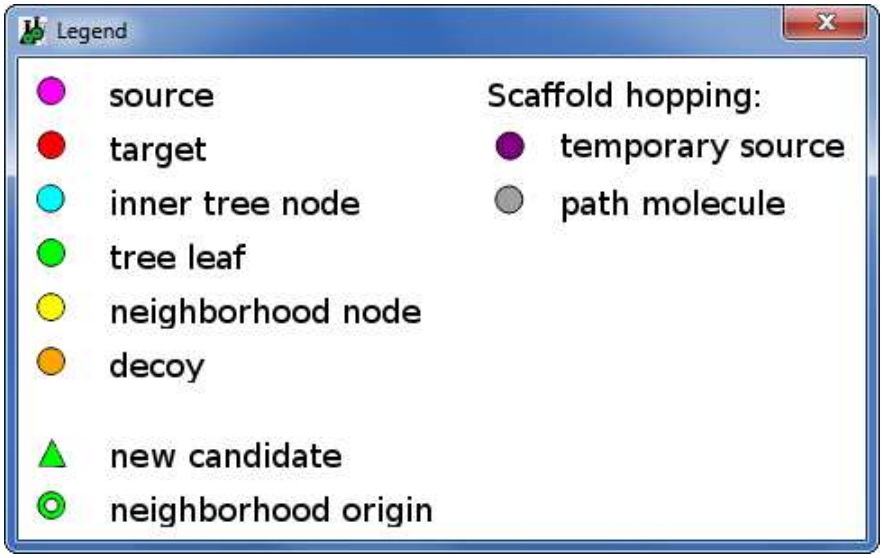
Obrázek 2.5: Dialog vytvoření jobu.

- Nastaví se zdrojová a cílová molekula společně s morfovacími operátory (v případě scaffold hoppingu se vybrané operátory budou používat na úrovni původních molekul), fingerprintem, podobnostní metodou, metodou pro redukci dimenze, parametry výpočtu atd. Mohou se nastavit i návnady (decoys), ale pro přehlednost zůstane jejich množina prázdná. Důležité je zaškrtnout checkbox "Scaffold hopping" (viz obrázek 2.5).

- Po potvrzení dialogu začne výpočet. Visualizace výpočtu se otevře přes dokovací část GUI "Job queue". V grafice je několik novinek oproti původní aplikaci. Tou hlavní novinkou jsou molekuly nových typů – dočasná zdrojová a cílová molekula a molekula výsledné cesty (viz obrázek 2.6).

- Zároveň se v levém horním rohu grafiky zobrazuje informace o aktuální scaffold úrovni ve tvaru "*textovy_popis (cislo_urovne/pocet_urovni)*". Celý vzhled grafického znázornění stavu výpočtu je vidět na obrázku 2.7.

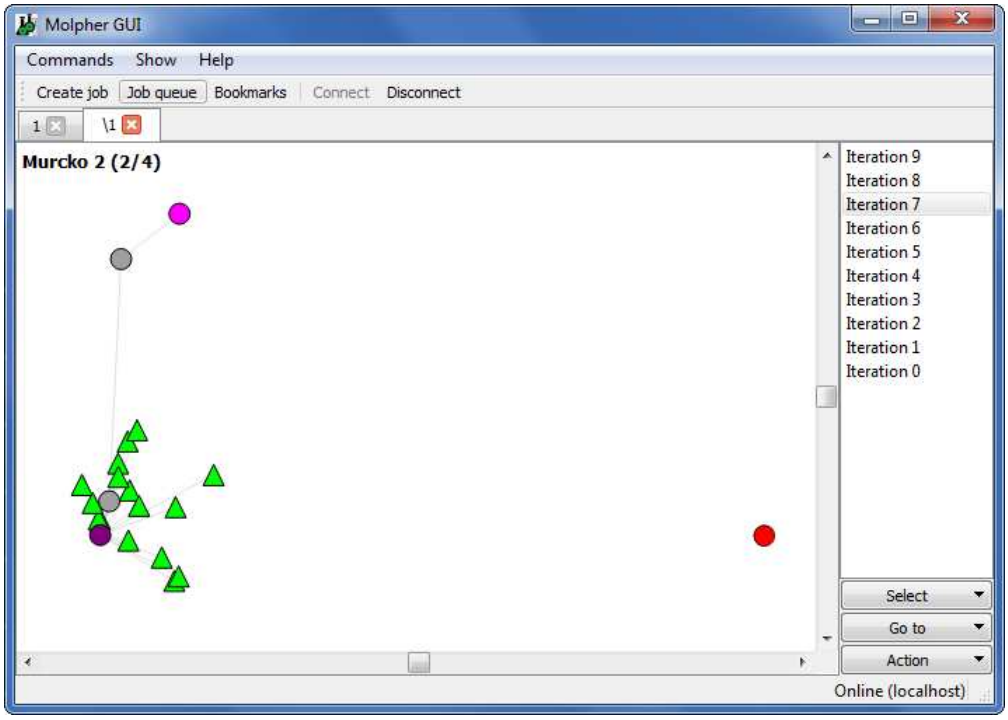
- Zbylá funkcionalita se aktualizovala a zůstala zachována. Jako příklad mohu zmínit zvýrazňování cesty z molekuly pod kurzorem myši do zdroje. Takže



Obrázek 2.6: Nové typy molekul jsou vidět v pravé části.

se zvýrazňují i molekuly z výsledné cesty mezi celkovým a dočasným zdrojem. Stejně tak posunutí grafiky na celkový zdroj/cíl funguje, i když jsou překryty těmi dočasnými.

Pro další informace o používání nebo technických detailech slouží stránka projektu Molpher [18], kde jsou nejen informace o scaffold hoppingu, které jsou obsažené v tomto dokumentu.



Obrázek 2.7: Zobrazení sedmé iterace při zapnuté technice scaffold hopping.

3. Testování

Součástí této diplomové práce je i experimentální ověření funkčnosti původního Molpheru bez techniky scaffold hopping a současné verze s touto technikou. Experimenty probíhaly ve třech oblastech – ověření kvality výsledků (měřeno vzdáleností nalezené cesty k jiným molekulám), testování rychlosti (počty iterací potřebných pro nalezení cest a celkový čas) a prozkoumání struktur výsledných molekul.

3.1 Nastavení Molpheru

Při experimentech se spouštěl jak Molpher odevzdaný v rámci softwarového projektu, tak aktuální verze s podporou techniky scaffold hopping. Parametry výpočtů pro všechny experimenty byly nastaveny co nejvíce stejně, aby se výsledky lišily pouze technikou scaffold hopping. Původní verze (odevzdaná jako SW projekt) je ale jiná a to v následujících bodech:

- *Redukce dimenze* – původní verze obsahovala tři metody redukce dimenze. Aktuální verze má jen dvě, přičemž v obou verzích je pouze metoda *Kamada-Kawai* (je nejnáročnější na výpočet a zároveň nejvíce vypovídající). V původní verzi pak byly ještě dvě rychlé metody – *Random*, která pro každou molekulu pouze vracela náhodnou pozici z předem dané čtvercové plochy, a *Dummy* vracející vždy polohu [0;0]. V aktuální verzi je již zmíněná metoda *Kamada-Kawai* a k ní přibyla metoda *PCA*. Redukční metoda *PCA* je dostatečně rychlá a proto bylo po jejím přidání rozhodnuto o odstranění výše zmíněných dvou metod z Molpheru. Metodu *Kamada-Kawai* jsem pro experimenty nevybral z důvodu její časové náročnosti (čas výpočtu by se prodloužil na dvojnásobek), i když byla jediná v obou verzích. Takže původní verze byla při testování spouštěna s *Dummy* metodou (měla stejné vlastnosti jako metoda *Random* a tudíž bylo jedno, která se vybere) a ta aktuální s *PCA*, což při testování na čas může, sice minimálně, ale přece, zhoršit výsledky aktuální verze.

- *Syntetizovatelnost* – toto je novinka, v původní verzi úplně chyběla. Nebyla přidána v rámci této diplomové práce, ale před ní. Při použití této možnosti se filtrují molekuly, které nejsou syntetizovatelné. V aktuální verzi je syntetizovatelnost defaultně nastavena a tak i zůstala při testování.

Zbylé nastavení výpočtu bylo stejné pro všechny tři způsoby spuštění. Pro úplnost tyto parametry uvedu na následujících řádcích:

- *Morfovací operátory* – byly vybrány všechny
- *Návnady* – množina zůstala prázdná
- *Fingerprint* – Morgan
- *Podobnostní metoda* – Tanimoto
- *Redukční (vizualizační) metoda* – Principal Component Analysis (PCA)

- Kandidát je blízko cíli, když je jeho vzdálenost menší než 0.1
- Kandidát, který není blízko cíli, může generovat maximálně 90 morfů za iteraci
- Kandidát, který je blízko cíli, může generovat maximálně 200 morfů za iteraci
- Molpher se pokusí přijmout alespoň 40 morfů za iteraci
- Molpher přijme maximálně 150 morfů za iteraci
- Molpher přijme morf, který má váhu alespoň 0 Daltonů
- Molpher přijme morf, který má váhu maximálně 500 Daltonů
- Molekula může vytvořit maximálně 5000 morfů za job
- Kandidát, který nezlepšuje cestu, přežije 6 iterací
- Cesta se hledá maximálně 1000 iterací
- Cesta se hledá maximálně 6000 hodin

3.2 Kvalita nalezených cest

Při testování kvality nalezených cest se testovala sada bioaktivních molekul, které jsou aktivní na určitých receptorech. Srovnávala se aktuální verze se zapnutou technikou scaffold hopping a Molpher bez této techniky odevzdaný jako SW projekt. Ze sady bioaktivních molekul se postupně vybíraly libovolné dvě molekuly, nalezla se cesta a zkoumalo se, jak je cesta podobná (jak je blízko) zbylým molekulám z dané množiny. Podobnost cesty se zbylými molekulami z množiny byla určena nejvyšší podobností mezi cestou a jakoukoliv molekulou z množiny (bez zdroje a cíle). Čím vyšší byla podobnost (tzn. menší vzdálenost), tím kvalitnější byla cesta.

Nebyl zde žádný předpoklad, že by Molpher s technikou scaffold hopping měl nalézat kvalitnější cesty, protože se hledají stejnými operátory, pomocí stejných fingerprintů i podobnostních metod. Tímto experimentem se tedy mělo ukázat, že Molpher s technikou scaffold hopping produkuje stejně kvalitní cesty.

Pro výše zmíněný účel byl navržen a implementován program, který je popsán v následující sekci[1].

3.2.1 Popis programu

V tomto odstavci je popsáno, jak probíhá otestování jedné cesty, což představuje jeden graf (grafy jsou zobrazeny dále v textu). Program potřebuje ke svému běhu předem vytvořený SNP soubor, kde je již nastavena zdrojová a cílová molekula z množiny bioaktivních molekul. Tento program spustí serverovou část Molpheru, počet spuštění n dostane jako argument, a v případě vygenerování každé jedné

[1] Je dostupný online na `https://xp-dev.com/wiki/185241/Homepage`

cesty se spočítají podobnosti cesty vůči všem ostatním molekulám z množiny. Po skončení n běhů se zjistí střední hodnoty a rozptyly podobností pro každou molekulu z množiny i nejbližší molekuly z cesty k ostatním molekulám z množiny v každém hledání cesty (tyto nejbližší molekuly mají informativní charakter a v grafech nejsou uvedeny).

Serverová část Molpheru musí být uložena na stejném počítači jako je program pro testování, protože se po ukončení výpočtu zpracovávají data o cestě, které server uložil na disk.

Struktura adresáře

Program předpokládá určitou strukturu adresáře, ve kterém je uložen. Každá množina bioaktivních molekul musí být uložena ve zvláštním adresáři, který je uložen na úrovní programu (vedle binárního souboru programu). Adresář s množinou bioaktivních molekul musí obsahovat jeden textový soubor ("input_smiles.txt") a alespoň jeden iniciální SNP soubor (oba soubory musí být ručně vytvořeny – textový soubor pomocí textového editoru a SNP soubor pomocí Molpheru):

- input_smiles.txt – obsahuje seznam molekul z množiny definovaných řetězcem SMILES. Na každém řádku je jedna molekula
- SNP soubory – jedná se o vstupní soubory pro serverovou část Molpheru. Každý SNP soubor má nastavenou zdrojovou a cílovou molekulu, fingerprint metodu, podobnostní metodu atd. Tyto čtyři vlastnosti jsou uložené v názvu každého SNP souboru. Ten má tvar **aa-bb_cc_dd.snp**:
  - *aa* ... pořadí vstupní molekuly v textovém souboru
  - *bb* ... pořadí výstupní molekuly v textovém souboru
  - *cc* ... zkratka fingerprintu
  - *dd* ... zkratka podobnostní metody

  Zkratky fingerprint metod jsou následující:

- ATP = Atom Pairs
- MRG = Morgan
- TOP = Topological
- TL1 = Topological Layered 1
- TL2 = Topological Layered 2
- TPT = Topological Torsion
- EATP = Atom Pairs (ext)
- EMRG = Morgan (ext)
- ETOP = Topological (ext)
- ETL1 = Topological Layered 1 (ext)

- ETL2 = Topological Layered 2 (ext)
- ETPT = Topological Torsion (ext)

Zkratky podobnostních metod mají tvar:

- ALB = All Bit
- ASM = Asymmetric
- BBL = Braun-Blanquet
- COS = Cosine
- DIC = Dice
- KLC = Kulczynski
- MCC = McConnaughey
- ONB = On Bit
- RSL = Russel
- SKL = Sokal
- TNM = Tanimoto
- TSB = Tversky (substructure)
- TSP = Tversky (superstructure)

Použití z příkazové řádky

Program je spouštěn z příkazové řádky a podporuje následující seznam argumentů:

- --help ... Zobrazení nápovědy.
- -M [--molpher-path] cesta ... Cesta k serverové části Molpheru (defaultně aktuální adresář).
- -I [--iterations] počet opakování ... Počet generovaných cest jedné konfigurace neboli jednoho SNP souboru (defaultně 5).
- -D [--del-temp-files] *true*/*false* ... Aktivuje/deaktivuje mazání výsledků generovaných serverovou částí Molpheru (defaultně *true*).

Pokud není uživatelem nastaven poslední argument, má implicitně hodnotu *true*. Je to z důvodu, aby nedocházelo k naplnění disku, protože výstupní soubory serverové části Molpheru mohou mít stovky megabytů i jednotky gigabytů v závislosti na počtu iterací. Kdyby se disk naplnil, Molpher by nespadl, ale ukládal by prázdné soubory. Testující program by pak tyto prázdné soubory načetl, vydal varovnou hlášku a danou konfiguraci ignoroval.

Výstupy

Pokud složka představující jednu množinu nemá odpovídající strukturu, je ignorována a pokračuje se další složkou. V opačném případě je spuštěna serverová část Molpheru, která začne hledat cesty pro jednotlivé iniciální SNP soubory. Výstupem jsou tři soubory s názvy začínajícími stejně jako iniciální snapshot ("aa-bb_cc_dd"):

- **aa-bb_cc_dd-similarities.csv** – obsahuje tabulku měření podobností případné nalezené cesty s molekuly z množiny
- **aa-bb_cc_dd-closest_path_mols.txt** – v tomto souboru je pro každé měření a každou molekulu z množiny (kromě zdrojové a cílové) uložena nejbližší molekula z cesty (v řetězci SMILES)
- **aa-bb_cc_dd-overview.txt** – zde je uložen souhrn informací o vzdálenostech molekul z množiny k cestám. Jako první jsou vypsány počty nalezených a nenalezených cest, a v přídadě, že se nalezla alespoň jedna cesta, jsou vygenerované střední hodnoty a rozptyly podobností pro každou molekulu z množiny (opět se netýká zdroje a cíle).

3.2.2 Vstupní data

Testovala se sada osmi bioaktivních molekul na estrogenních receptorech [8, 27]. V následujícím seznamu jsou vypsaná jejich ID (označené jako CID – Compound Identification Number) z PubChem databáze:

1. CID 698
2. CID 102614
3. CID 5280961
4. CID 6095481
5. CID 448537
6. CID 5035
7. CID 2733526
8. CID 104741

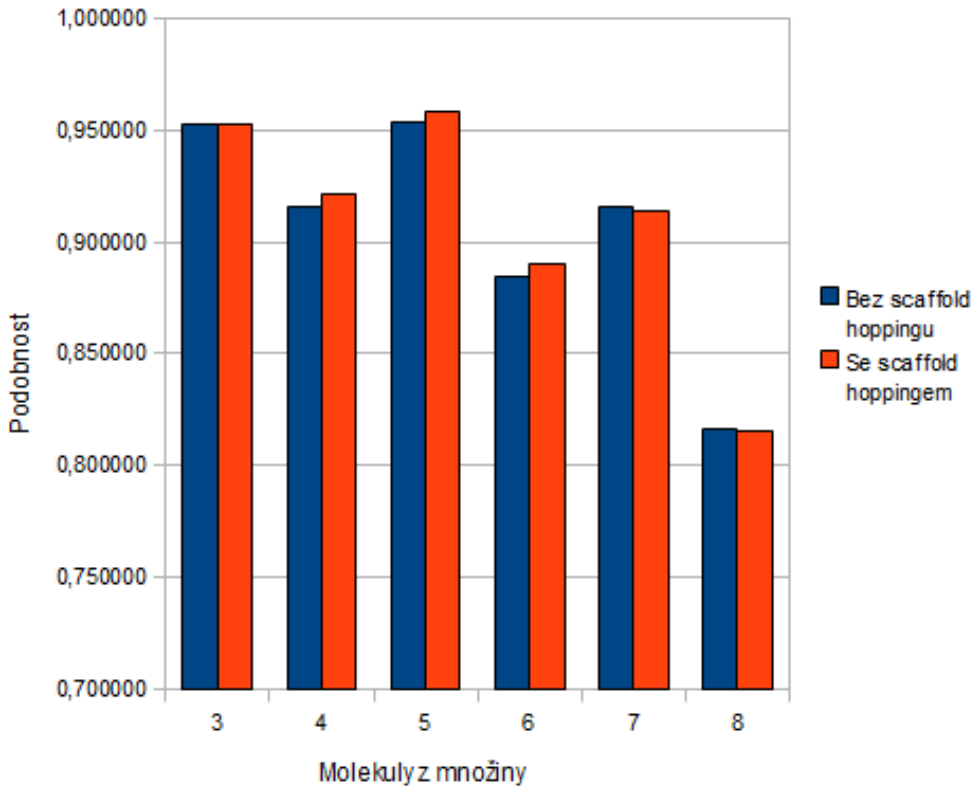
3.2.3 Výsledky

Výsledky tohoto experimentu jsou ve formě grafů a jedné tabulky, ve které jsou aritmetické průměry středních hodnot a rozptylů. Na konci této sekce jsou výsledky slovně popsány.

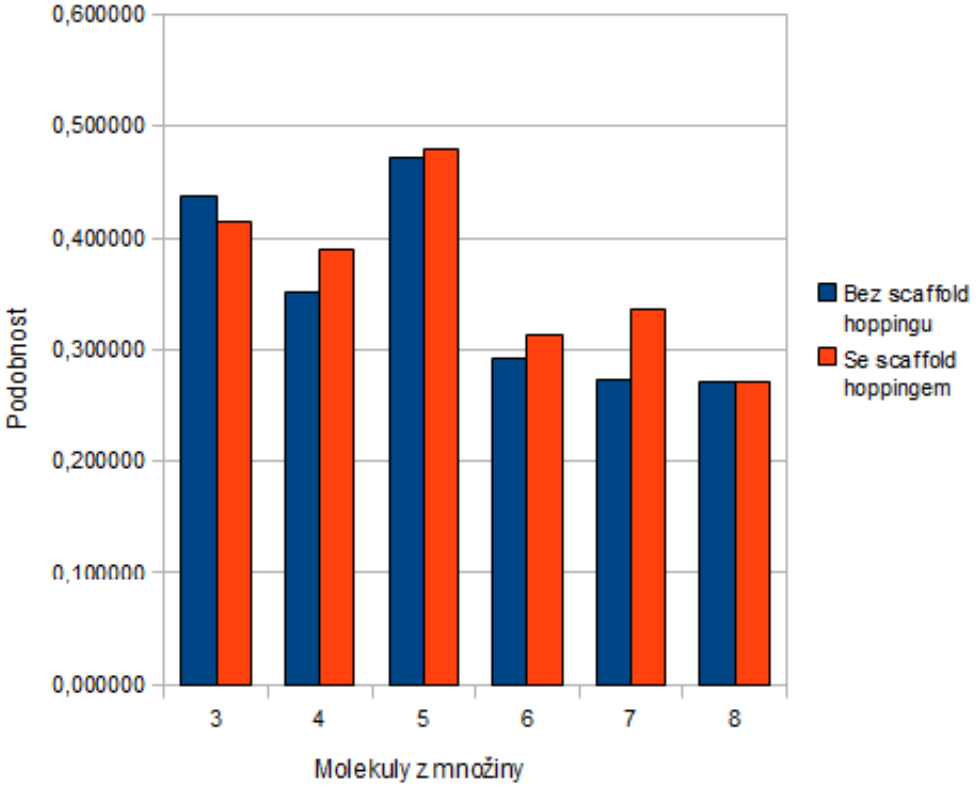
Z titulku každého grafu lze vyčíst použitý fingerprint a podobnostní metodu (význam textu v titulku je stejný jako jméno SNP souboru bez přípony, což je popsáno v sekci 3.2.1). Joby byly nastaveny defaultně (více o nastavení viz podkapitola 3.1 – fingerprinty a podobnostní metody se při testování kvality nalezených cest mění).

Každé generování cesty (představující jednu barvu sloupečků grafu) bylo spuštěno **10x** a z těchto deseti výsledků se počítala střední hodnota a rozptyl. Pro jeden graf se tedy desetkrát spustil původní Molpher bez techniky scaffold hopping a desetkrát s ní. Na grafech, kromě dvou posledních, jsou znázorněny střední hodnoty podobností molekul z dané množiny vůči nalezeným cestám. Protože množina obsahovala osm molekul a libovolné dvě byly vybrány jako zdroj a cíl (konkrétně první molekula jako zdroj a druhá molekula jako cíl nebo naopak), ke zbylým šesti se znázornily podobnosti. Poslední dva grafy ukazují průměrný počet iterací při hledání cest – každá dvojice sloupečků odpovídá jednomu grafu podobností.

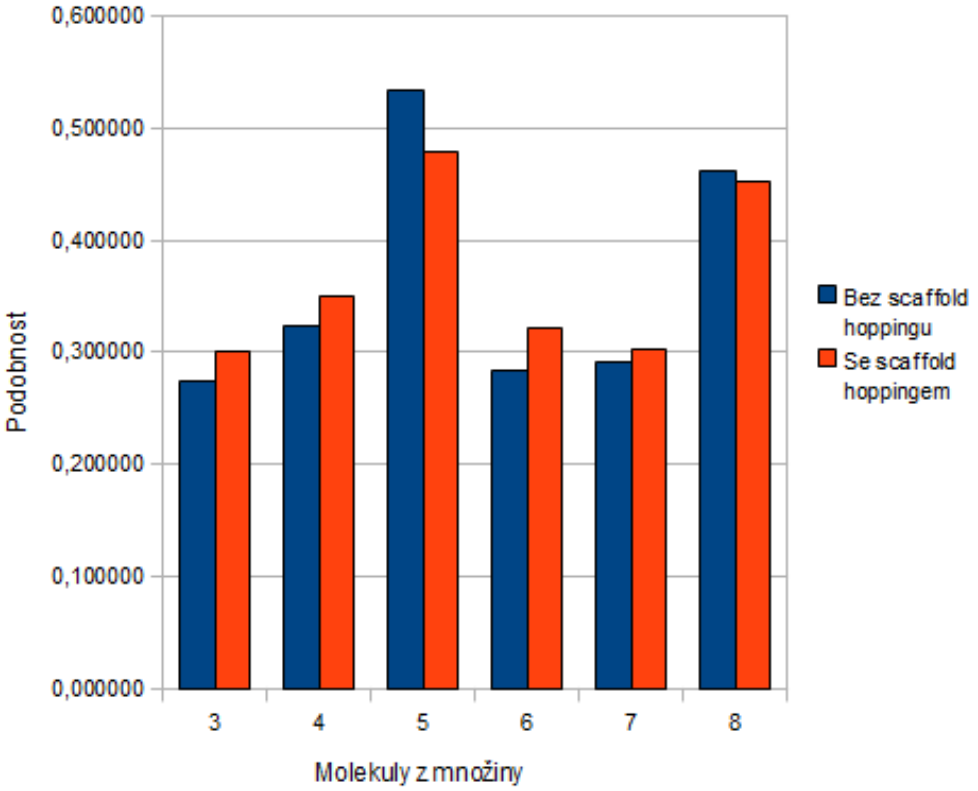
Graf číslo 3.1 ukazuje, jak jsou molekuly číslo tři až osm z dané množiny v průměru podobné nalezené cestě, kde jako zdroj byla vybrána první molekula z množiny a jako cíl druhá molekula z množiny. Fingerprint byl nastaven na *Atom Pairs* a podobnostní metoda byla *All Bit* (viz sekce 3.2.1 pro význam popisku grafu). S technikou scaffold hopping byly blíže nalezené cestě v průměru čtyři molekuly ze šesti (třetí až šestá molekula). Zbylé dvě molekuly ze šesti byly blíže nalezené cestě s použitím Molpheru bez techniky, nicméně ve všech šesti případech jsou rozdíly malé. Na ostatních grafech jsou výsledky podobné – podobnosti se liší jen minimálně.



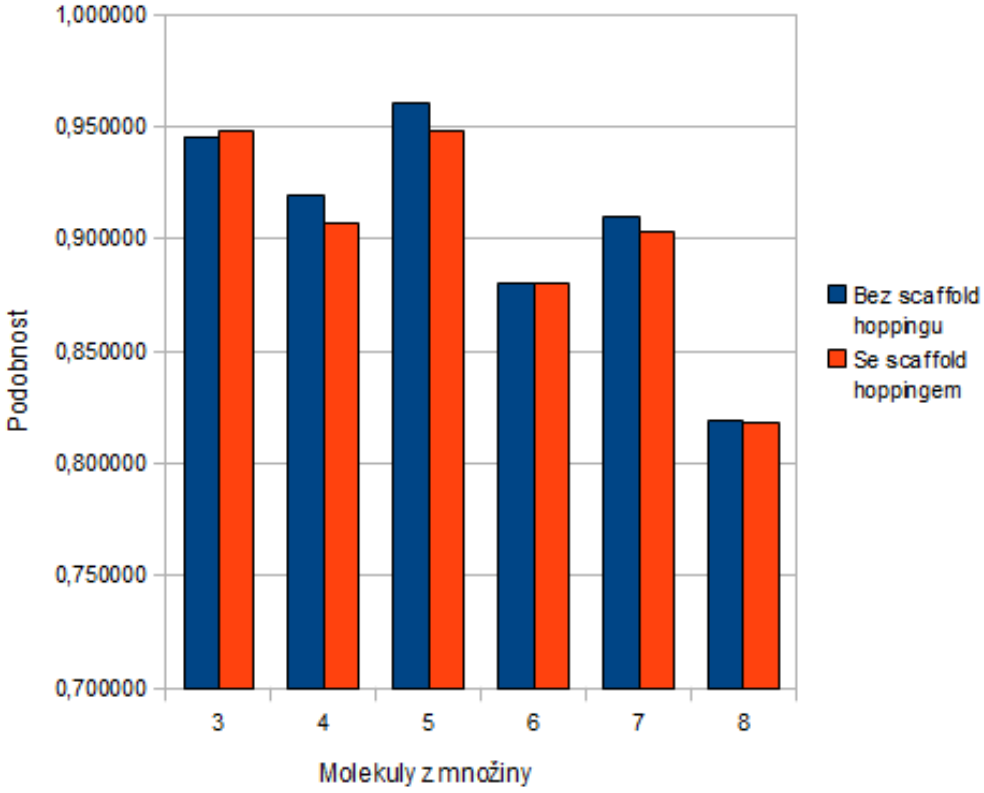
Obrázek 3.1: 1-2_ATP_ALB.



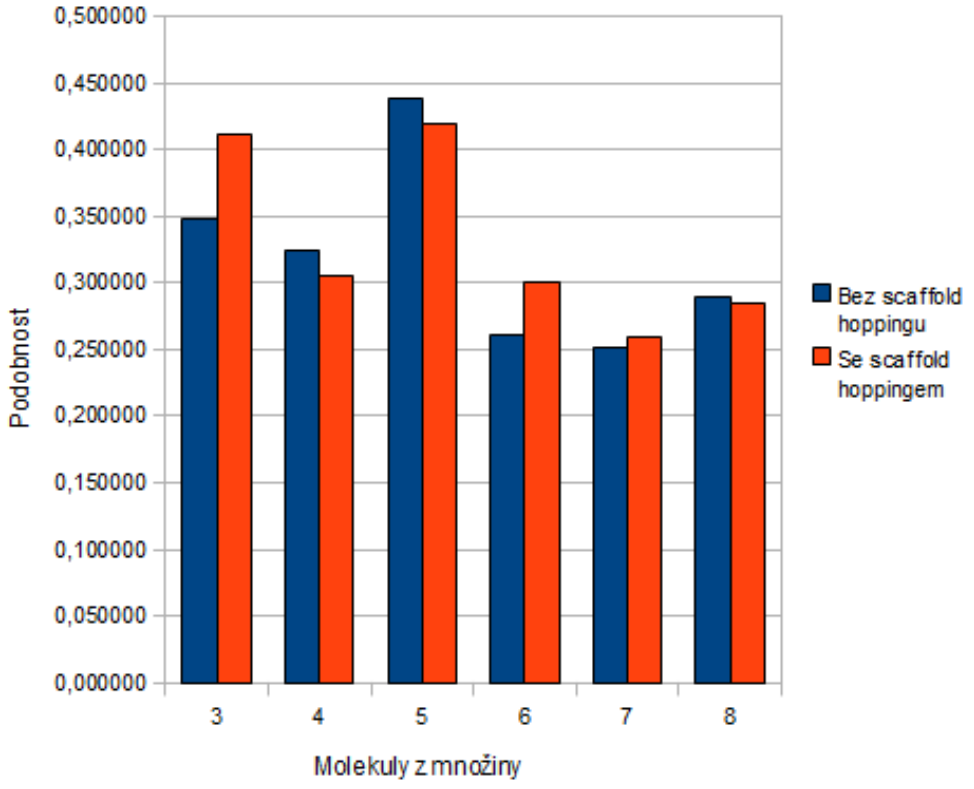
Obrázek 3.2: 1-2_ATP_TNM.



Obrázek 3.3: 1-2_MRG_TNM.

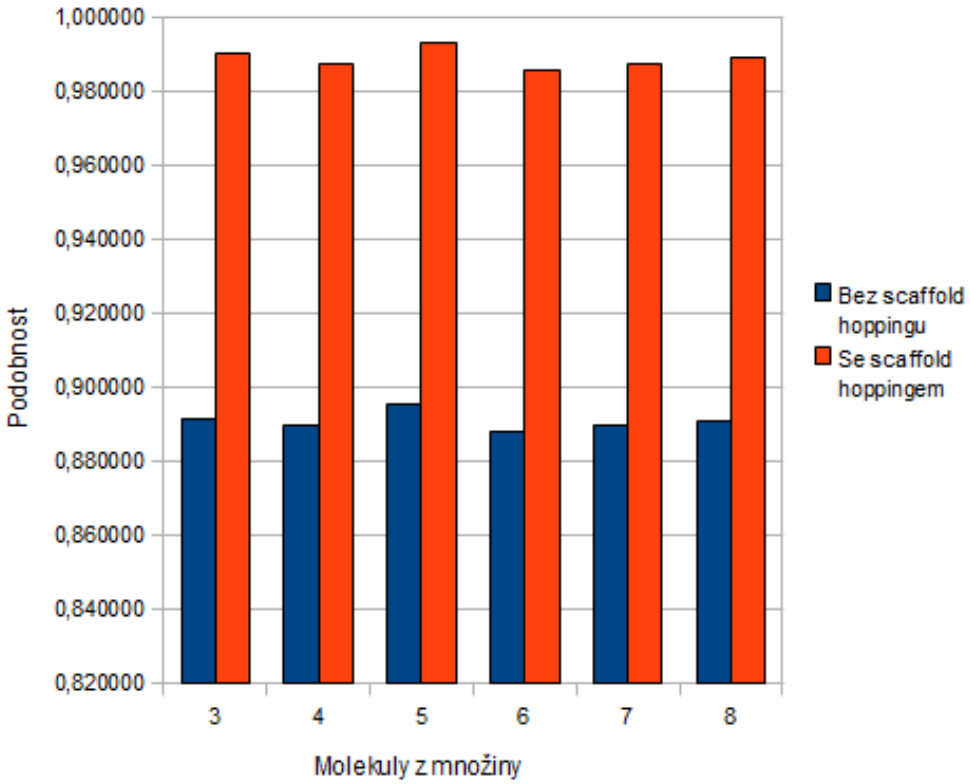


Obrázek 3.4: 2-1_ATP_ALB.

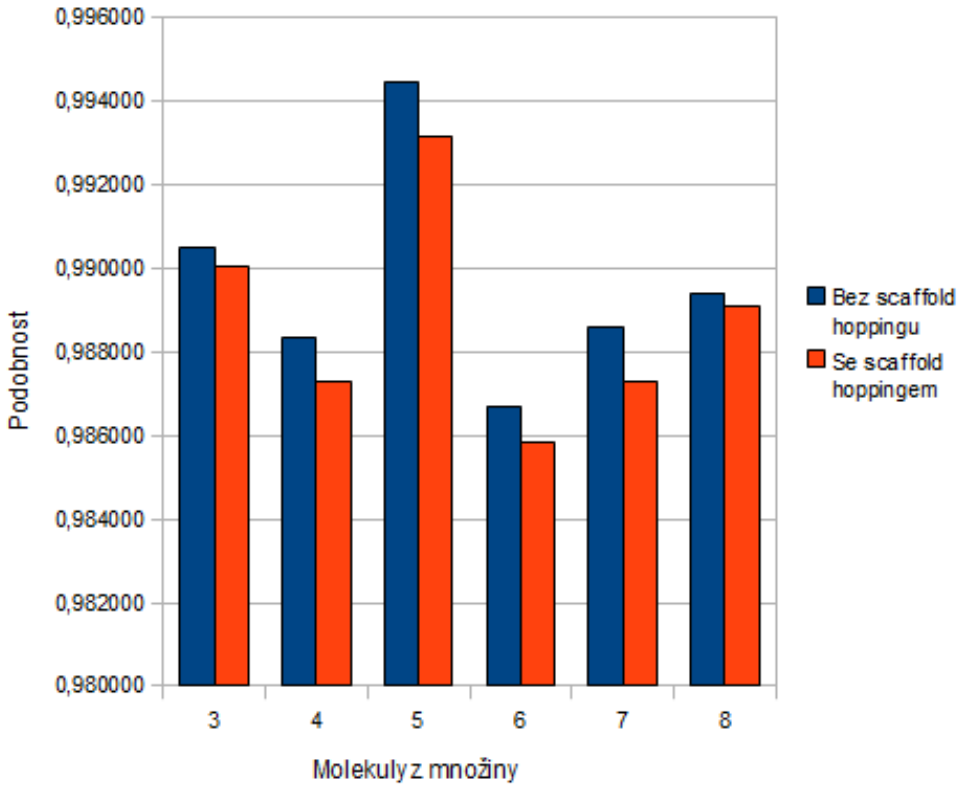


Obrázek 3.5: 2-1_ATP_TNM.

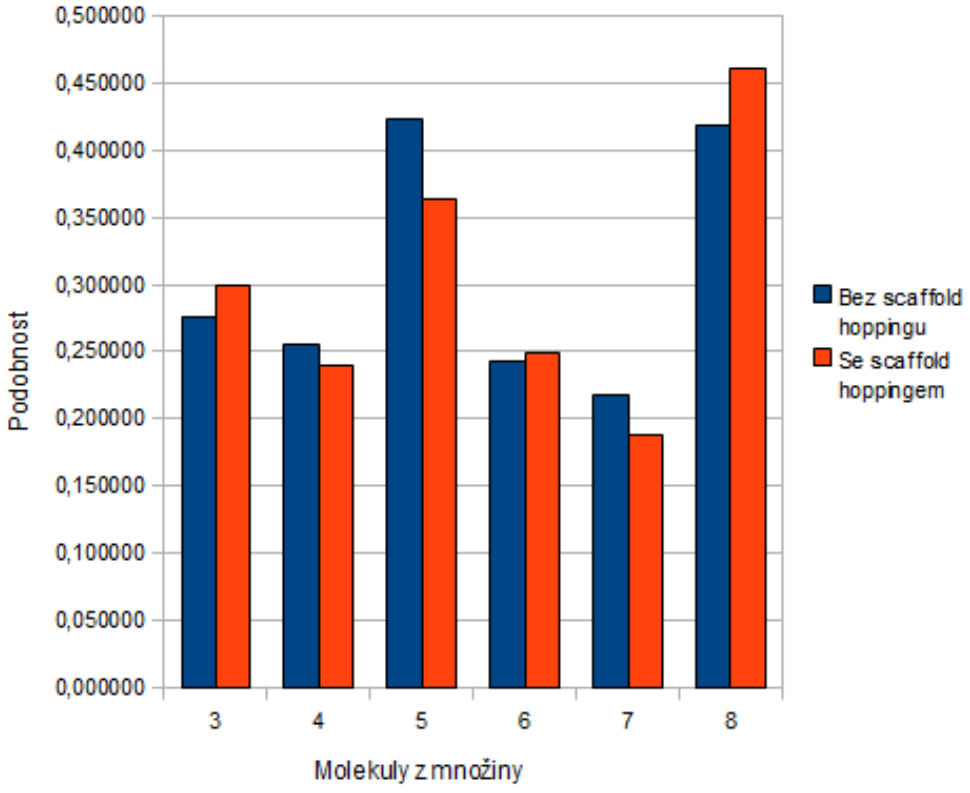
Výsledky měření z grafů 3.6 a 3.8 jsou ovlivňené skutečností, že v obou případech Molpher bez techniky scaffold hopping jednou neskončil. Je to vidět především na prvním grafu, protože tam byly hodnoty podobností vysoké a tudíž započítání nulové podobnosti snížilo střední hodnotu znatelněji. Očištěná data bez započítání nenalezených cest jsou na grafech 3.7 a 3.9.



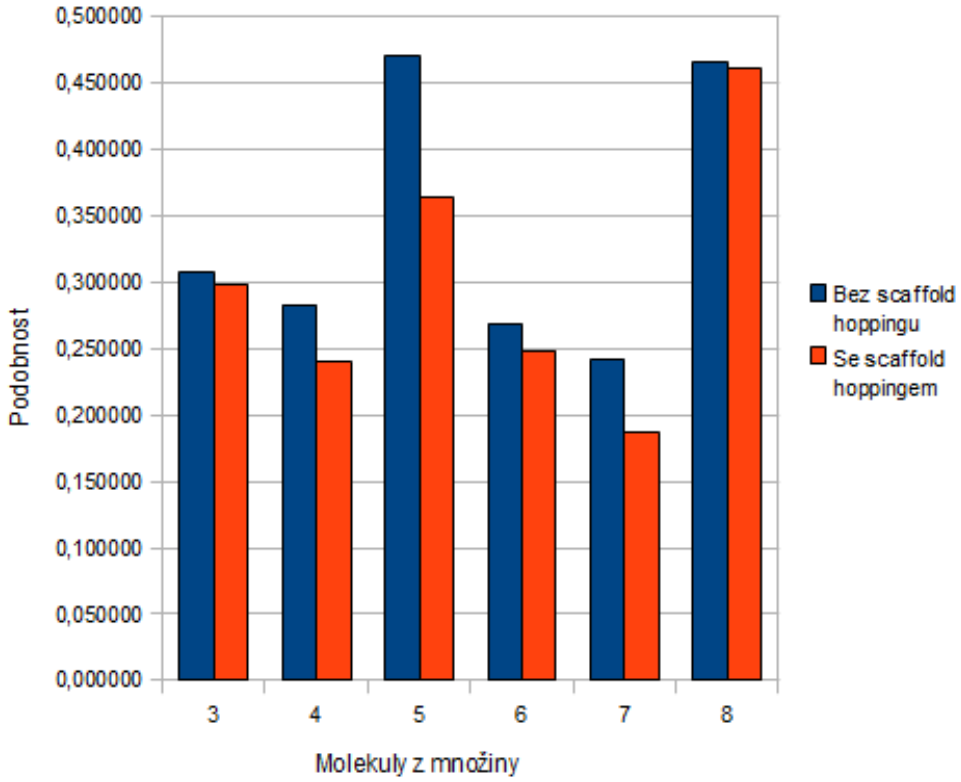
Obrázek 3.6: 2-1_MRG_ALB.



Obrázek 3.7: 2-1_MRG_ALB – očištěná data.



Obrázek 3.8: 2-1_MRG_TNM.



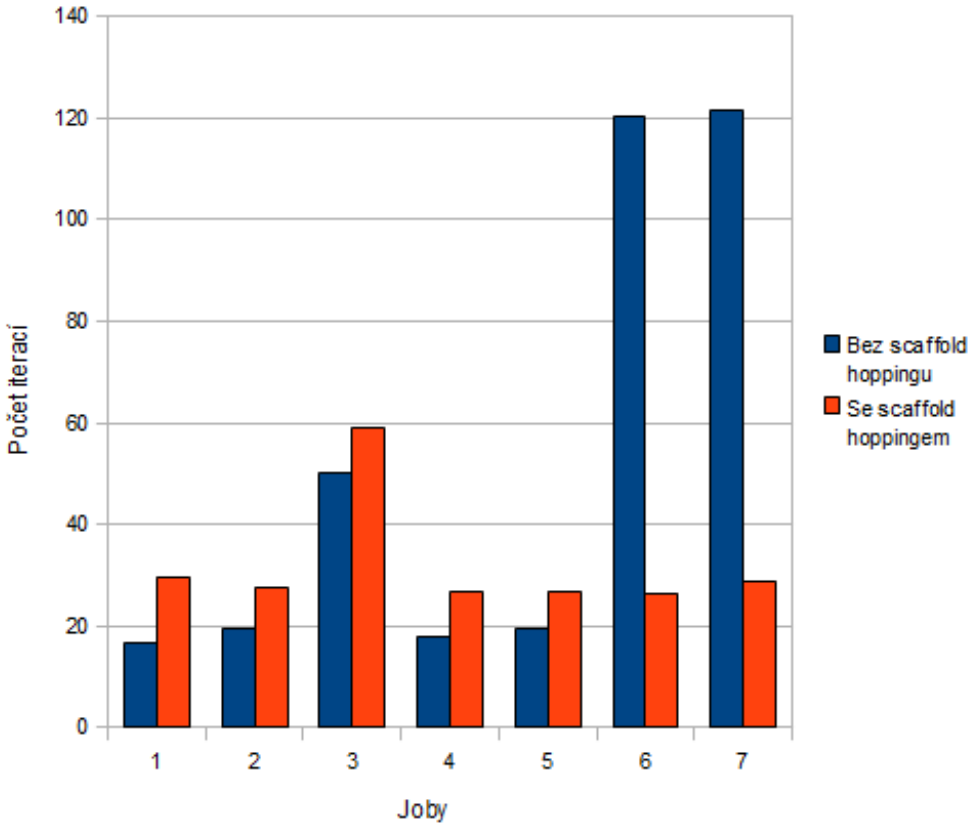
Obrázek 3.9: 2-1_MRG_TNM – očištěná data.

Poslední dva grafy číslo 3.10 a 3.11 ukazují průměrné počty kroků jednoho jobu, což představuje hledání jedné cesty s danou konfigurací. Každý job odpovídá jednomu grafu zobrazenému výše v této sekci. První graf číslo 3.10 zobrazuje naměřené výsledky bez očištění od extrémů (s nenalezenými cestami). Druhý graf již je očištěný od extrémů.

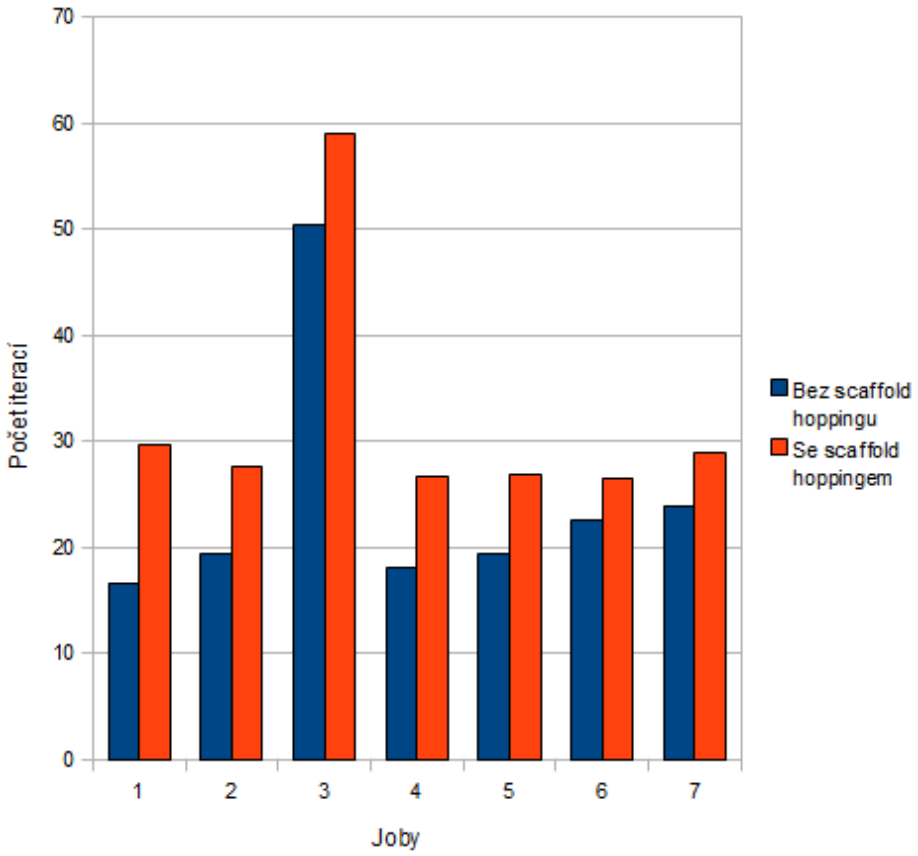
Joby v následujících grafech jsou číslovány postupně podle předešlého zobrazení grafů podobností. Tedy:

- 1 . . . 1-2_ATP_ALB
- 2 . . . 1-2_ATP_TNM
- 3 . . . 1-2_MRG_TNM
- 4 . . . 2-1_ATP_ALB
- 5 . . . 2-1_ATP_TNM
- 6 . . . 2-1_MRG_ALB
- 7 . . . 2-1_MRG_TNM

U posledních dvou jobů v grafu číslo 3.10 jsou průměrné počty iterací bez techniky scaffold hopping vyšší, protože v obou případech jednou výpočet neskončil a bylo tedy započítáno celých tisíc kroků výpočtu.



Obrázek 3.10: Průměrná délka výpočtu.



Obrázek 3.11: Průměrná délka výpočtu.

Z grafu číslo 3.11 vyplývá, že při kratších cestách (do šedesáti iterací) potřebuje Molpher se zapnutou technikou scaffold hopping více kroků. To není nic překvapivého, jelikož na obecnějších úrovních mohou být molekuly stejné, přepne se výpočet do další úrovně a tím je započítaný jeden krok, i když se jen přepnula scaffold úroveň. Pro delší cesty – bude zkoumáno v další sekci – je tu předpoklad, že nakonec bude celkový počet iterací menší.

	Průměrná střední hodnota	Průměrný rozptyl
Bez scaffold hoppingu	0.5909	0.00068
Se scaffold hoppingem	0.5948	0.00035

Tabulka 3.1: Aritmetické průměry střední hodnoty a rozptylu.

Shrnutí výsledků

V této části věnované prozkoumání kvality generovaných výsledků bylo testováno sedm různých konfigurací Molpheru. Pro každý z těchto sedmi případů byla připravena množina osmi molekul z nichž se vybral zdroj a cíl a zjišťovala se vzdálenost nalezené cesty k šesti zbylým molekulám (nejkratší vzdálenost mezi molekulou ze šesti zbylých a molekulami na cestě). V prvních sedmi grafech jsou na y-ové ose znázorněny podobnosti, což má přímou souvislost se vzdáleností (obě veličiny jsou v rozmezí $< 0; 1 >$ a platí, že součet podobnosti a vzdálenosti je přesně jedna).

Z grafů lze vyčíst, že naměřené hodnoty jsou velmi podobné a nelze tedy jednoznačně říci, že Molpher s technikou scaffold hopping nalézá kvalitnější cesty či naopak. To potvrzuje i tabulka 3.1, kde aritmetický průměr střední hodnoty je vyšší s technikou scaffold hopping jen o tisíciny. S touto novou funkcionalitou jsou tedy výsledky stejně kvalitní. Průměrný rozptyl je s technikou scaffold hopping poloviční oproti Molpheru bez techniky, což znamená, že technika scaffold hopping dává stabilnější výsledky (nejsou tak velké rozdíly v podobnostech).

3.3 Počty kroků a čas

Při tomto experimentu byla zkoumána kvalita Molpheru z pohledu rychlosti – po kolika iteracích a za jak dlouho se nalezne cesta. Aby výsledky byly co nejvíce vypovídající, jedna konfigurace (kde zdrojová a cílová molekula se nemění) byla spuštěna původním Molpherem odevzdaným jako SW projekt, pak aktuální verzí Molpheru bez zapnuté techniky scaffold hopping a do třetice také nejnovější verzí, ale se zapnutou technikou scaffold hopping.

Grafy s výsledky časů jsou oddělené od sumarizace počtu iterací, nicméně jsou přehledně za sebou na jedné stránce a je vidět kolik času průměrně stojí dané počty iterací. Když hodnota na y-ové ose je menší nebo rovna jedné a zároveň se jedná o logaritmickou stupnici s minimální hodnotou jedna, sloupeček není vidět. Celkem bylo otestováno 24 dvojic zdroj-cíl. Každý graf obsahuje šest dvojic a tudíž jsou zde vidět čtyři grafy zobrazující počty iterací (to stejné platí i pro grafy ukazující časové hodnoty). Každá dvojice má své ID, které se používá v grafech dále v této sekci. Seznam všech 24 dvojic je následující (ve tvaru "*ID. zdroj – cíl*"):

1. CID 4550 – CID 3140
2. CID 8711 – CID 7747
3. CID 7956 – CID 7913
4. CID 8099 – CID 6518
5. CID 6219 – CID 3042
6. CID 5094 – CID 3276
7. CID 9679 – CID 2574
8. CID 6168 – CID 3454
9. CID 9679 – CID 4938
10. CID 8393 – CID 3242
11. CID 4983 – CID 2570
12. CID 5719 – CID 5522
13. CID 9362 – CID 5955
14. CID 6211 – CID 5781
15. CID 4358 – CID 3344
16. CID 5842 – CID 5838
17. CID 5843 – CID 2481
18. CID 6464 – CID 4099
19. CID 4990 – CID 1018
20. CID 8557 – CID 8515
21. CID 3969 – CID 2467
22. CID 9689 – CID 6142
23. CID 5537 – CID 3280
24. CID 9077 – CID 7097

Všechny grafy, kromě generování cest třináct až osmnáct, mají logaritmickou y-ovou osu, protože obsahují jak nízké, tak i vysoké hodnoty lišící se v řádech

(kvůli nenalezení cest). Zmíněná výjimka (týká se jak počtu iterací, tak času) má lineární osu y, jelikož všechny cesty byly nalezeny a hodnoty tudíž nejsou tak vysoké.

Každá jednotlivá konfigurace byla spuštěna **10x** a následně se z naměřených výsledků spočítal průměr počtu iterací a časů. Tento způsob z daných tří testování byl časově nejnáročnější, jelikož se pro jednu dvojici zdroj-cíl spouštěly tři různé konfigurace Molpheru desetkrát. Následné ruční zpracování si také vyžádalo další nemalou časovou režii.

Z grafů je patrné, že ve více případech nová verze Molpheru (nezáleží na tom, zda byla zapnutá technika scaffold hopping či nikoliv) nalezla cestu zatímco program odevzdaný jako softwarový projekt skončil neúspěšně. Týká se to především první a čtvrté dvojice (*ID 1* a *ID 4*) v grafech 3.12 a 3.13. Mezi mojí úpravou a verzí vzniklou SW projektem se také modifikovala aplikace. Zkoumáním jsem zjistil, že již Molpher před přidáním techniky scaffold hopping vykazoval lepší výsledky. Tudíž tento stav není mnou zapříčiněný. Pravděpodobně to bude způsobené kombinací úprav – přidání filtrace pomocí syntetizovatelnosti atd. Opírám se i o dodatečné měření první dvojice (*ID 1*), kde původní Molpher desetkrát z deseti spuštění neskončil a novější Molpher před samotným přidáním techniky scaffold hopping vygeneroval cestu ve čtyřech pokusech z pěti. Domnívám se tedy, že některé molekuly nový Molpher zahazuje, tím pádem se nevydává špatnou cestou a tudíž končí nalezením cesty rychleji.

V grafech 3.12 a 3.13 je vidět abnormalita. Touto zvláštností je cesta s *ID 3*, když byla spuštěna aktuální verze Molpheru se zapnutou technikou scaffold hopping (žlutý sloupeček). Průměrný počet iterací je 1000, takže všech deset běhů skončilo neúspěchem. V kontrastu s tím je ale průměrný čas maximálně jedna vteřina (v grafu 3.13 ani není vidět). Je to způsobené tím, že cílová molekula je tak jednoduchá, že pro ní je *Oprea 1* scaffold prázdná molekula. Mezi libovolnou molekulou a prázdnem se nenalezla cesta a těchto tisíc iterací skončilo záhy po necelé sekundě.

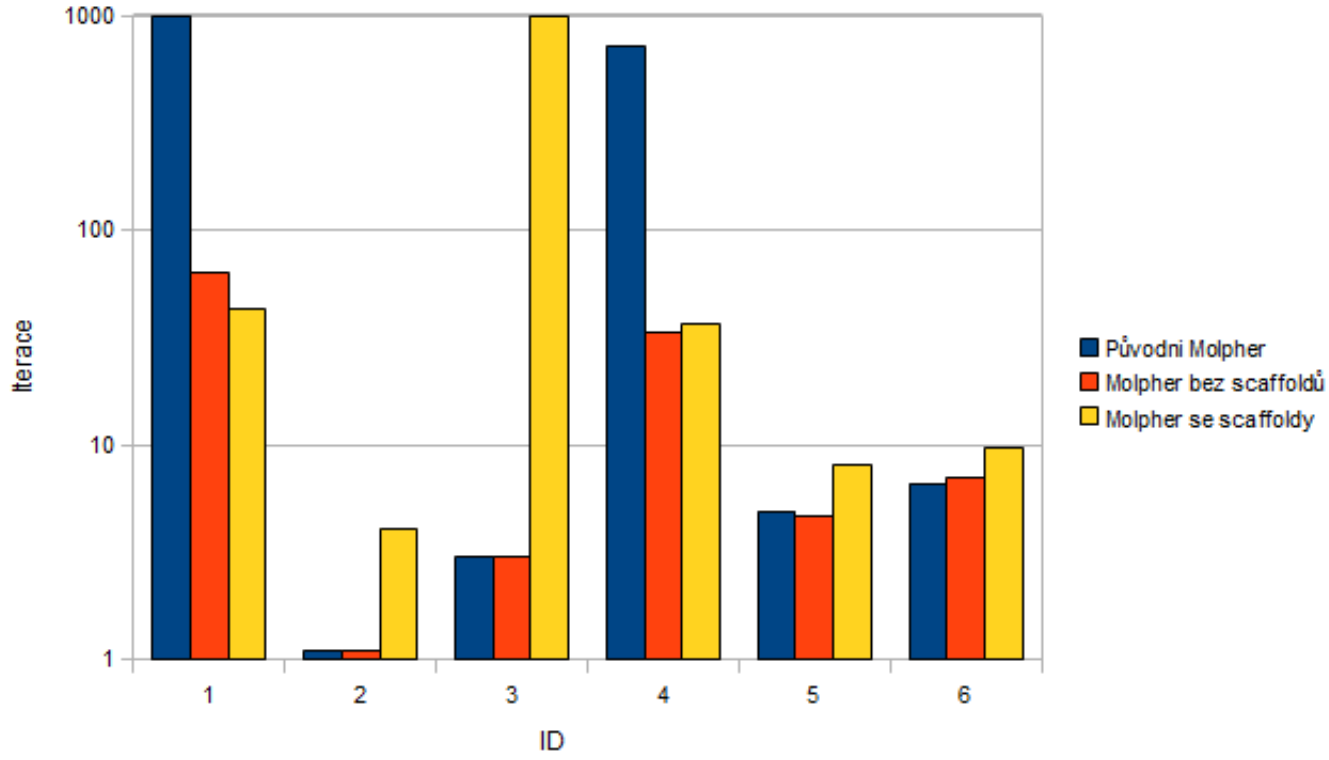
Hledání cest z grafu 3.15 trvalo, dle průměrných hodnot, pokaždé nejdéle Molpheru se zapnutou technikou scaffold hopping, i když ne vždy skončil po nejvyšším počtu iterací. Do počtu kroků trvaly nejdéle výpočty cest s *ID 9* a *12*. U první jmenované cesty byl nejrychlejší Molpher s technikou scaffold hopping, v případě druhé cesty byly hodnoty kroků přibližně stejné.

Cesty *13* až *18* byly krátké a tudíž grafy 3.16 a 3.17 nemají logaritmickou y-ovou osu. Při těchto krátkých cestách je technika scaffold hopping horší v počtu iterací i ve srovnání na čas.

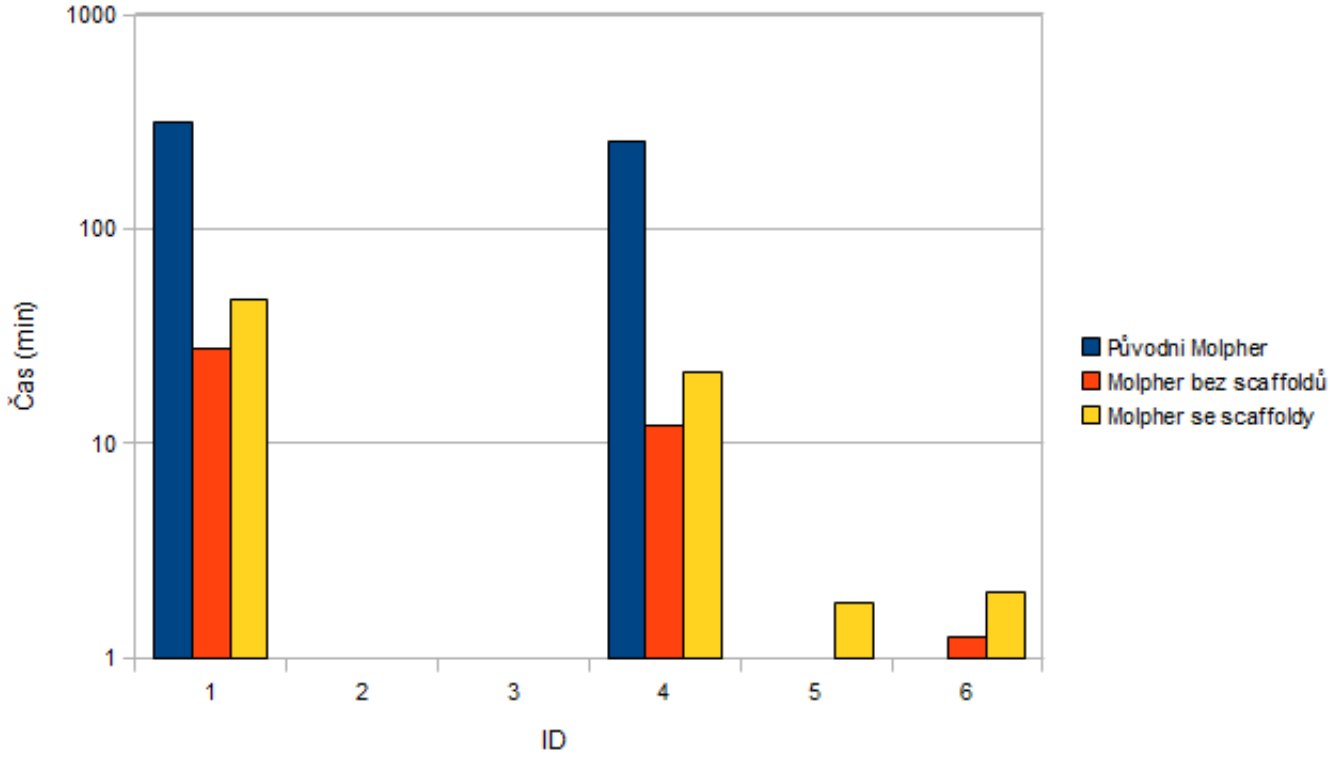
Poslední dvojice grafů s čísly 3.18 a 3.19 obsahuje tři delší cesty (delší cestou je myšlen počet iterací vyšší než 100 u alespoň dvou verzí Molpheru ze tří) – *ID 20*, *22* a *23*. Opět ve většině (dvě ze tří) z těchto cest má nejnižší počet iterací Molpher s technikou scaffold hopping.

Jak již bylo zmíněno výše v textu, z grafů a dodatečného ověření vyplynulo, že v některých případech je nejnovější verze kvalitnější v tom smyslu, že skončí o dost dříve než původní verze odevzdaná jako SW projekt. Dále vyplývá, že Molpher se zapnutou technikou scaffold hopping je časově pomalejší. Je to z toho důvodu, že při zapnuté technice se pracuje na dvou úrovních (původní molekula i scaffold), což znamená dvojnásobné vytváření RDKit molekul, jejich modifikace a režie. Existuje poměrně málo delších cest, které mají průměrnou délku mezi sto

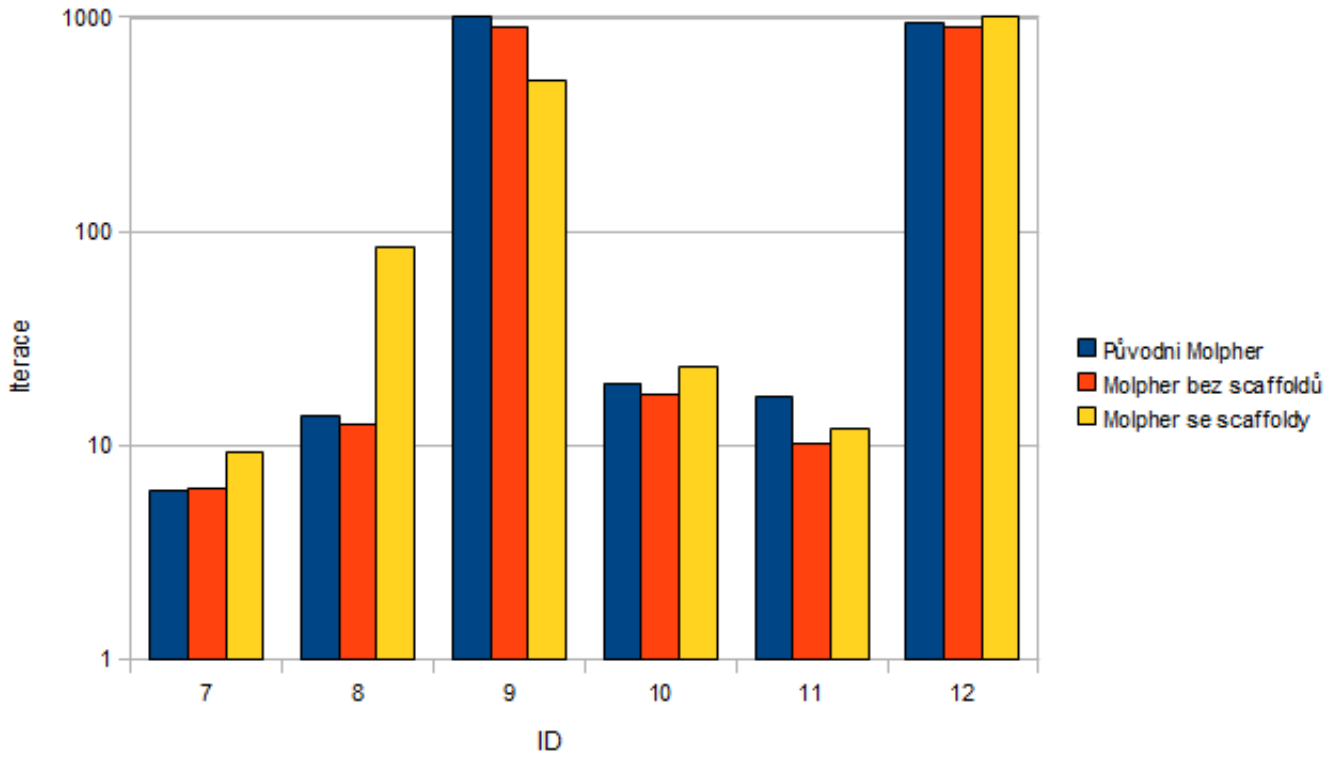
a tisíci kroky (tisíc kroků byla hranice, kdy Molpher skončil s hledáním), kde by se projevila výhoda techniky a cesta by se nalezla po méně krocích než bez scaffold hoppingu. Jedná se o cesty s *ID 9*, *12*, *20*, *22* a *23*, na nichž je vidět tendence menších počtů iterací s technikou scaffold hopping. Kdyby se maximální počet kroků zvýšil, bylo by lépe vidět, že při delších cestách je technika v počtu iterací rychlejší, ale již toto omezení na tisíc kroků bylo časově hodně náročné.



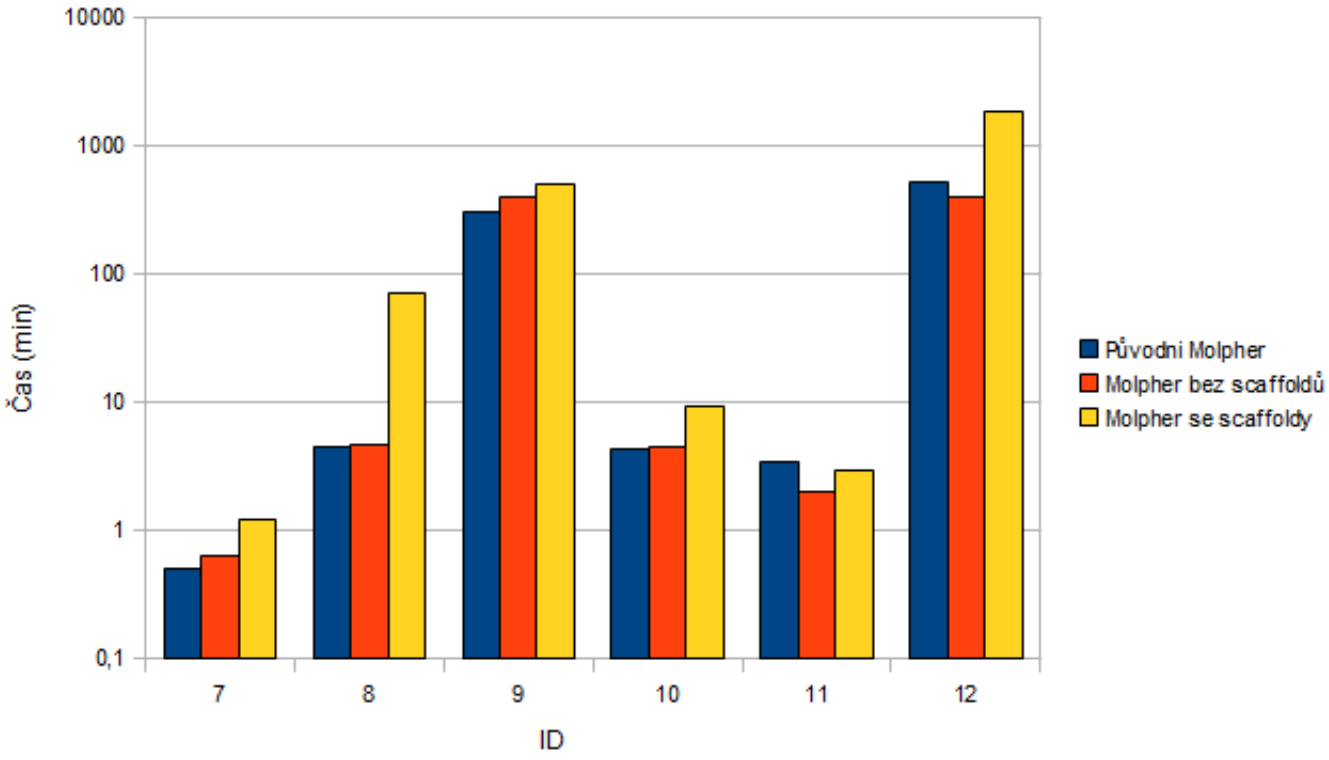
Obrázek 3.12: Srovnání počtu iterací (šestice cest).



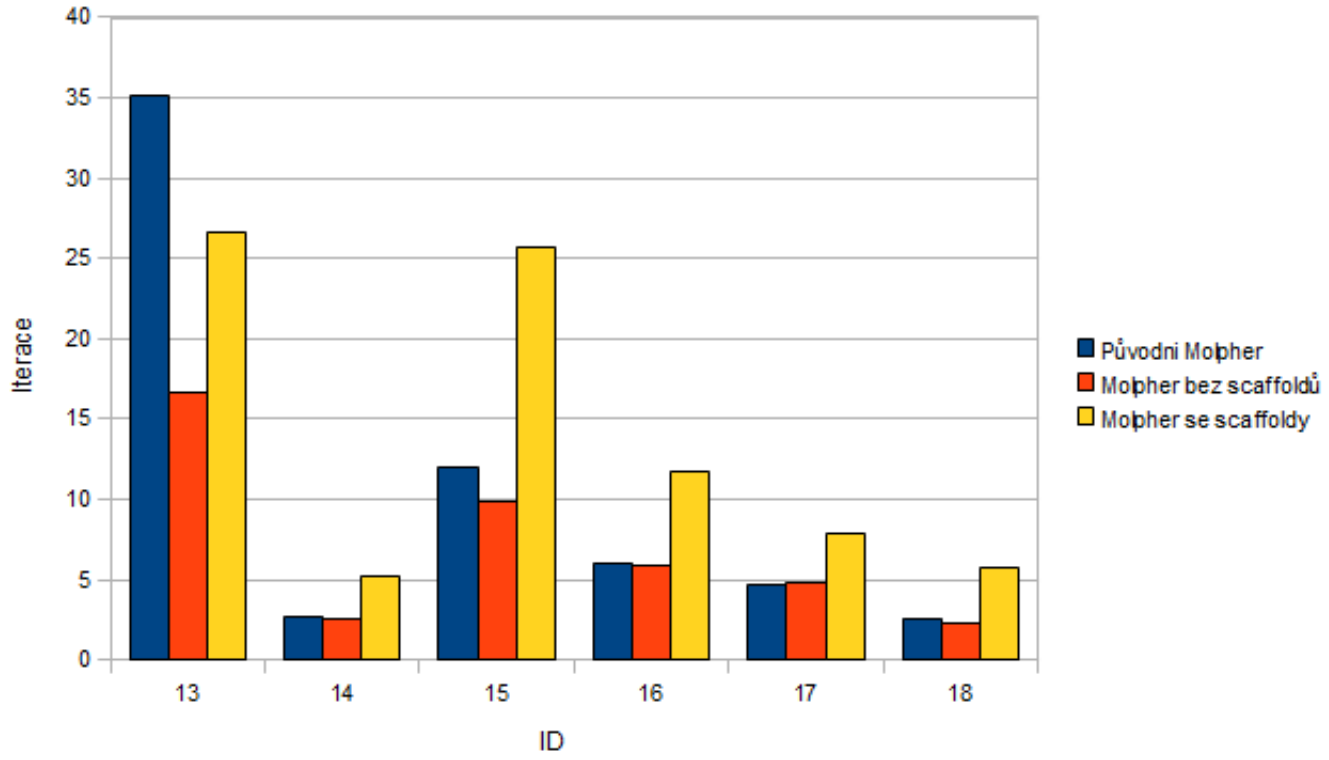
Obrázek 3.13: Srovnání časů (šestice cest).



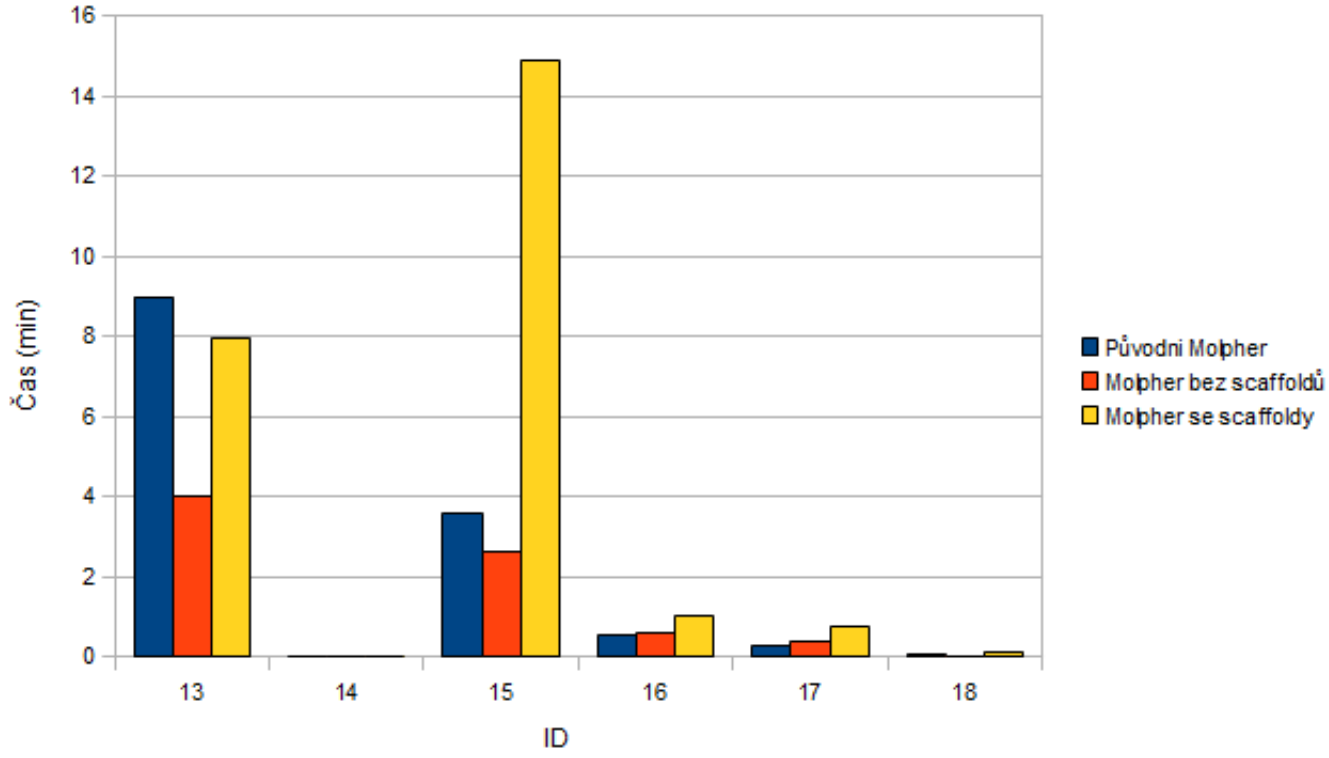
Obrázek 3.14: Srovnání počtu iterací (šestice cest).



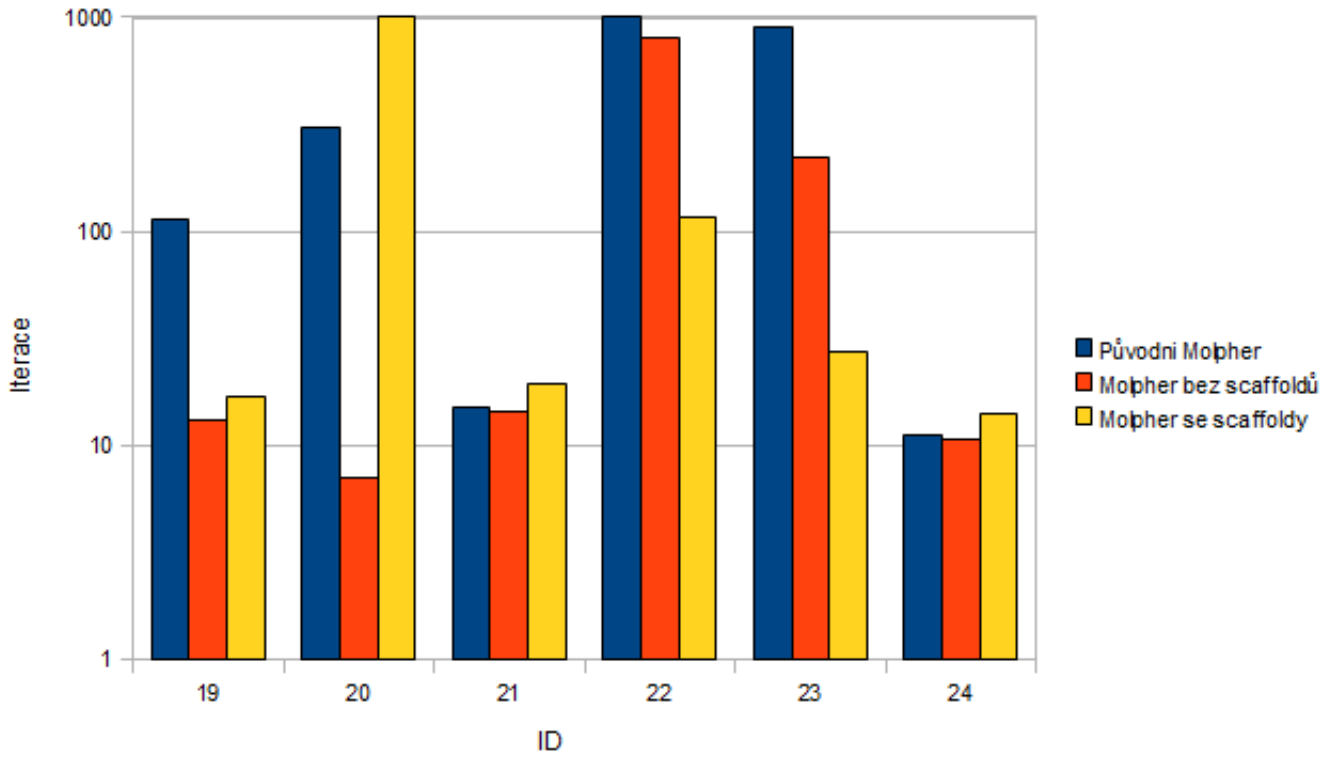
Obrázek 3.15: Srovnání časů (šestice cest).



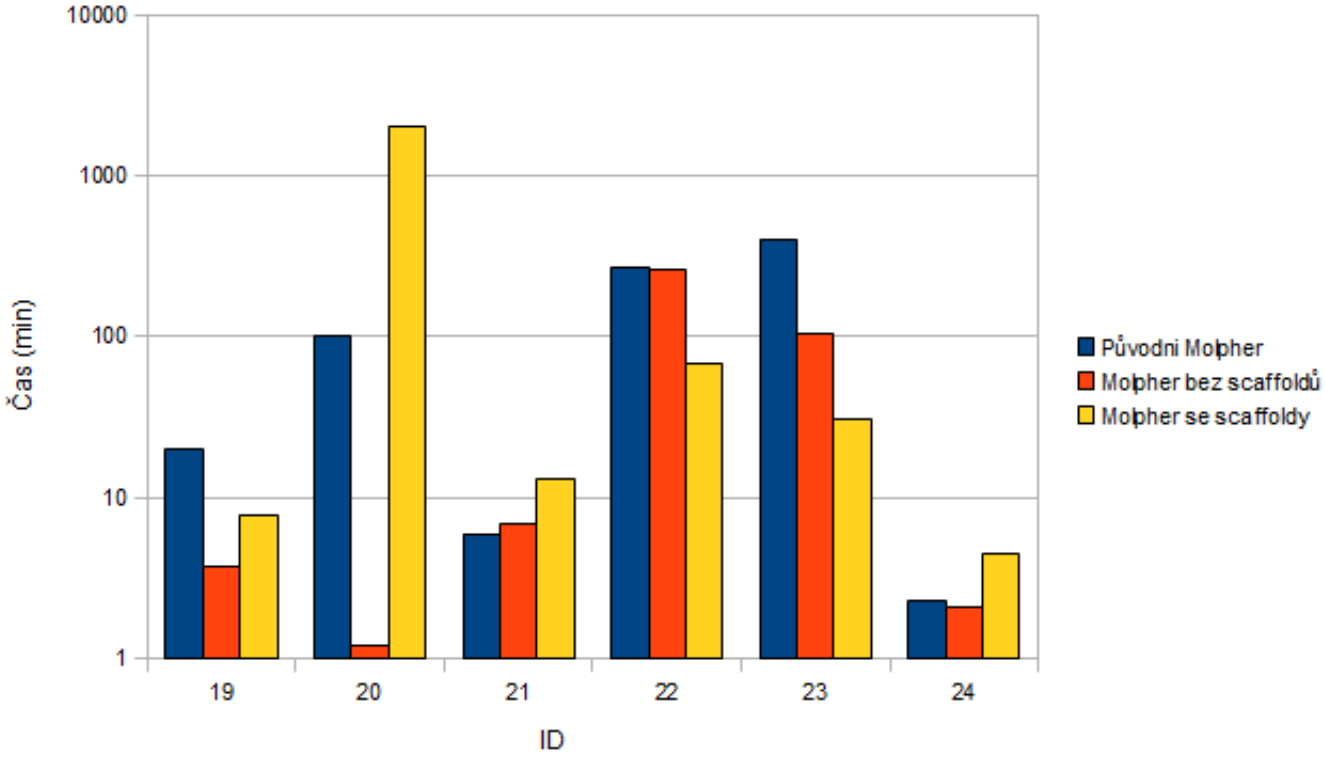
Obrázek 3.16: Srovnání počtu iterací (šestice cest).



Obrázek 3.17: Srovnání časů (šestice cest).



Obrázek 3.18: Srovnání počtu iterací (šestice cest).



Obrázek 3.19: Srovnání časů (šestice cest).

3.4 Strukturní rozmanitost

V této části se zkoumaly struktury všech molekul na dvojicích cest – jedna nalezená bez techniky scaffold hopping a druhá s ní. Technika scaffold hopping ve vyšších úrovních abstrakce pracuje především s kruhy a cestami mezi nimi, což jsou takzvaná jádra molekul. Byl zde tedy předpoklad, že díky této technice budou výsledné molekuly, co se týče jejich struktur, rozmanitější. Tato vlastnost by byla výhodná v drug discovery procesu.

Bylo prozkoumáno 20 párů cest, ze kterých se zjišťovaly počty všech typů scaffoldů. Tyto hodnoty se navzájem porovnávaly a kvantifikovaly. Procentuální celkové výsledky jsou v tabulce 3.2. Na konci této sekce o strukturní rozmanitosti jsou uvedeny dva konkrétní příklady, kdy Molpher s technikou scaffold hopping vygeneroval cesty s více scaffoldy než bez techniky scaffold hopping. Níže je seznam všech 20 různých cest určených dvojicemi *zdroj – cíl* (v prvním sloupci jsou krátké cesty, které obsahují méně než dvacet molekul a ve druhém sloupci jsou dlouhé cesty s alespoň dvaceti molekulami):

Krátké cesty	Dlouhé cesty
CID 8711 – CID 7747	CID 4550 – CID 3140
CID 6219 – CID 3042	CID 8099 – CID 6518
CID 5094 – CID 3276	CID 9679 – CID 4938
CID 6168 – CID 3454	CID 8393 – CID 3242
CID 4983 – CID 2570	CID 9362 – CID 5955
CID 6211 – CID 5781	CID 4358 – CID 3344
CID 5842 – CID 5838	CID 4990 – CID 1018
CID 5843 – CID 2481	CID 8557 – CID 8515
CID 6464 – CID 4099	CID 9689 – CID 6142
CID 3969 – CID 2467	CID 5537 – CID 3280

Vždy se porovnávaly stejně dlouhé cesty a zjišťovalo se, zda cesta vygenerovaná technikou scaffold hopping obsahuje více, stejně nebo méně scaffoldů než ta druhá vygenerovaná bez použití techniky scaffold hopping. Výsledky všech dvaceti testů jsou v tabulce 3.2, částečné výsledky podle délek cest (bylo testováno deset a deset cest) jsou v tabulkách 3.3 a 3.4.

Z tabulky vyplývá, že výsledné cesty s technikou scaffold hopping neobsahují vždy více různých scaffoldů než ty bez ní (platí hlavně pro krátké cesty). Při testování nastaly i případy, kdy Molpher s vypnutou danou technikou generoval rozmanitější molekuly než při zapnuté technice. Celkově je ale pravděpodobnější, že Molpher se zapnutou technikou scaffold hopping bude generovat rozmanitější molekuly než naopak (viz tabulka 3.2 – procenta na prvním řádku určující pravděpodobnost vyššího výskytu různých scaffoldů s technikou scaffold hopping jsou vyšší než procenta na posledním řádku). V případě delších cest jsou rozdíly v počtu různých scaffoldů mezi cestami markantnější, což je vidět při srovnání tabulek 3.3 a 3.4, kde se při delších cestách pravděpodobnost zvyšuje ve prospěch

techniky scaffold hopping.

	Oprea 1	Murcko 2	Rings with linkers 1	Původní molekuly
Více scaffoldů	25 %	35 %	40 %	0 %
Stejně scaffoldů	60 %	55 %	50 %	100 %
Méně scaffoldů	15 %	10 %	10 %	0 %

Tabulka 3.2: Srovnání počtu scaffoldů s technikou scaffold hopping.

	Oprea 1	Murcko 2	Rings with linkers 1	Původní molekuly
Více scaffoldů	20 %	20 %	30 %	0 %
Stejně scaffoldů	70 %	70 %	60 %	100 %
Méně scaffoldů	10 %	10 %	10 %	0 %

Tabulka 3.3: Srovnání počtu scaffoldů s technikou scaffold hopping pro cesty menší než 20 molekul.

	Oprea 1	Murcko 2	Rings with linkers 1	Původní molekuly
Více scaffoldů	30 %	50 %	50 %	0 %
Stejně scaffoldů	50 %	40 %	40 %	100 %
Méně scaffoldů	20 %	10 %	10 %	0 %

Tabulka 3.4: Srovnání počtu scaffoldů s technikou scaffold hopping pro cesty od 20 molekul.

Příklad 1

Na obrázcích 3.20 a 3.21 jsou vidět dvě stejně dlouhé cesty s tím rozdílem, že první byla vygenerována bez a druhá s technikou scaffold hopping. V obou případech je zobrazeno třináct molekul cesty. Při zapnuté technice se ale vygenerovalo více různých scaffoldů (viz tabulka 3.5).

V tomto příkladu hledaná cesta se zapnutou technikou obsahuje o jeden scaffold typu *Rings with linkers 1* více než druhá bez ní. Na obrázku číslo 3.20 se jedná o 1., 2., 4., 7. a 9. molekulu. Na druhém obrázku s číslem 3.21 jde o 1., 2., 3., 4., 5. a 7. sloučeninu.

Jak je vidět z tabulky, počty různých scaffoldů dvou nejobecnějších úrovní jsou stejné. V případě úrovně *Oprea 1* se jedná o dva nestejné scaffoldy (pokaždé první dvě molekuly cesty), u *Murcko 2* jsou to čtyři (na obrázku číslo 3.20 to jsou molekuly s čísly 1, 2, 7 a 9, v případě obrázku 3.21 je řeč o 1., 2., 3. a 4. molekule). V tabulce je zmíněna i poslední nejobecnější úroveň samotných molekul, kde každá molekula cesty je jiná a tudíž číslo třináct odpovídá počtu všech molekul na cestě.

Molpher programově rozlišuje molekuly na základě jejich SMILES, což je textový řetězec. Platí ale, že pro jednu molekulu může existovat více SMILES řetězců. Tudíž Molpher může mít v jeden okamžik ve stromu kandidátů dvě stejné

	Oprea 1	Murcko 2	Rings with linkers 1	Původní molekuly
Scaffold hopping vypnutý	2	4	5	13
Scaffold hopping zapnutý	2	4	6	13

Tabulka 3.5: Počty různých scaffoldů.

molekuly uložené pod jinými řetězci SMILES. Tato situace nastala i při zjišťování počtu různých scaffoldů typu *Murcko 2* na cestě vygenerované technikou scaffold hopping. Molekuly číslo 4 a 7 (obrázek 3.21) mají stejný scaffold, nicméně vygenerovaly se dva různé řetězce SMILES. Jednoznačné určení řetězce a molekuly, kde jedné sloučenině odpovídá jeden řetězec, existuje a nazýva se kanonický SMILES. Zda by se měla používat tato reprezentace nebo aktuální je otázkou, která však není součástí této diplomové práce.

Pro doplnění popíši jednu transformaci molekuly na jinou. Jedná se o poslední krok mezi předposlední molekulou a cílem na obrázku 3.21, kde byl použit operátor *Bond Reroute* (zkráceně BR). Na první pohled nemusí být srozumitelné jaká vazba se kam přepojila, proto se to pokusím objasnit. Jediný rozdíl mezi těmito dvěma molekulami je ve dvou atomech (tj. O a CH_3 v případě předposlední molekuly) v horní části levé "slepé cesty". Odpojila se vazba směřující do jádra molekuly od atomu O na straně atomu O a následně se přichytila k atomu CH_3. Touto změnou vnikla cílová molekula.

Příklad 2

Z tabulky číslo 3.6 je vidět, že v tomto příkladu má cesta se zapnutou technikou scaffold hopping pro změnu o jeden *Oprea 1* scaffold více než bez ní. V případě zapnuté techniky (obrázek 3.23) se jedná o 1., 3. a 4. molekulu cesty, které jsou různé na dané úrovni scaffoldů. A když se zaměříme na druhou vygenerovanou cestu s vypnutou technikou (obrázek 3.22), uvidíme, že různé *Oprea 1* scaffoldy jsou na 1., 2., 3. a 4. místě. Počty zbylých scaffoldů na těchto dvou cestách jsou stejné.

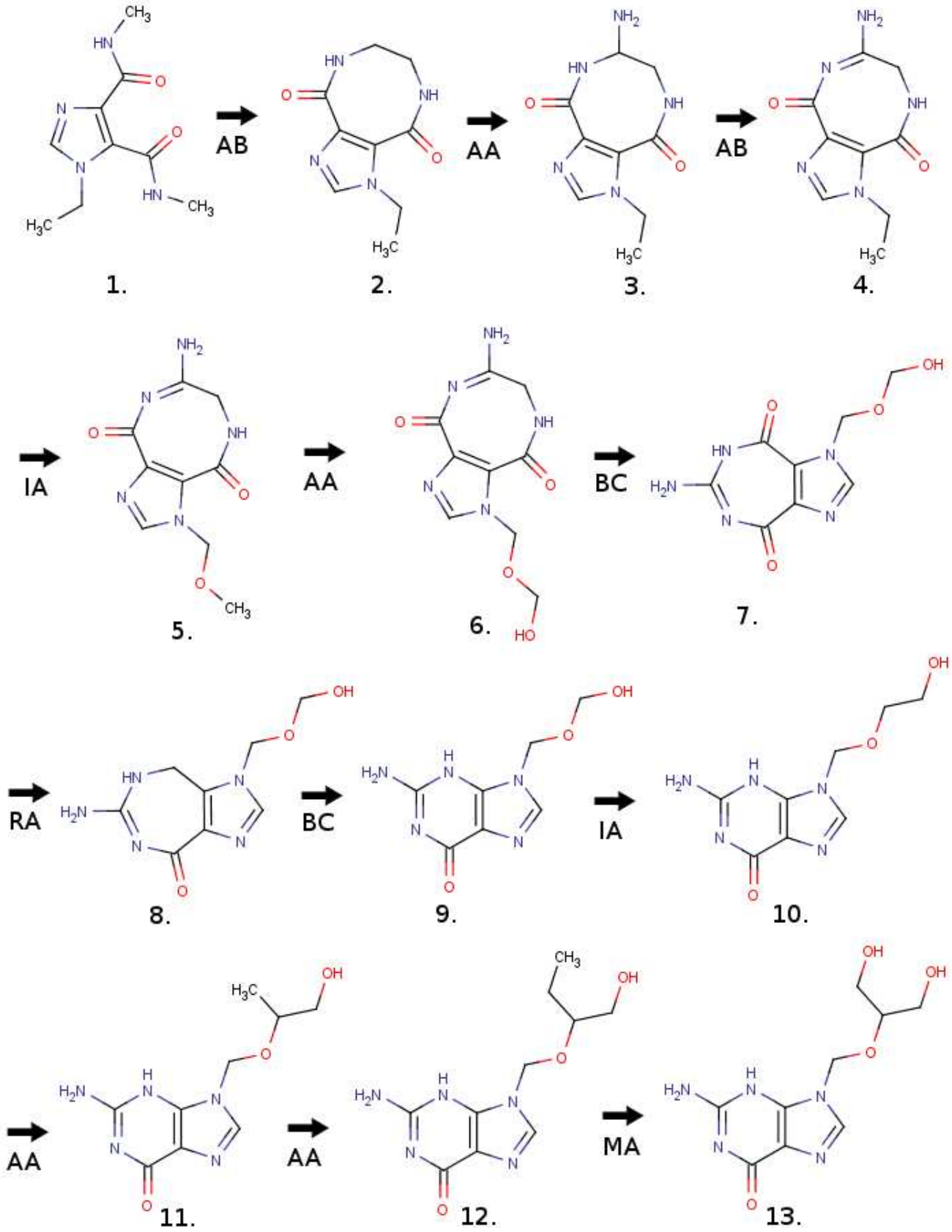
	Oprea 1	Murcko 2	Rings with linkers 1	Původní molekuly
Scaffold hopping vypnutý	3	12	14	17
Scaffold hopping zapnutý	4	12	14	17

Tabulka 3.6: Počty různých scaffoldů.

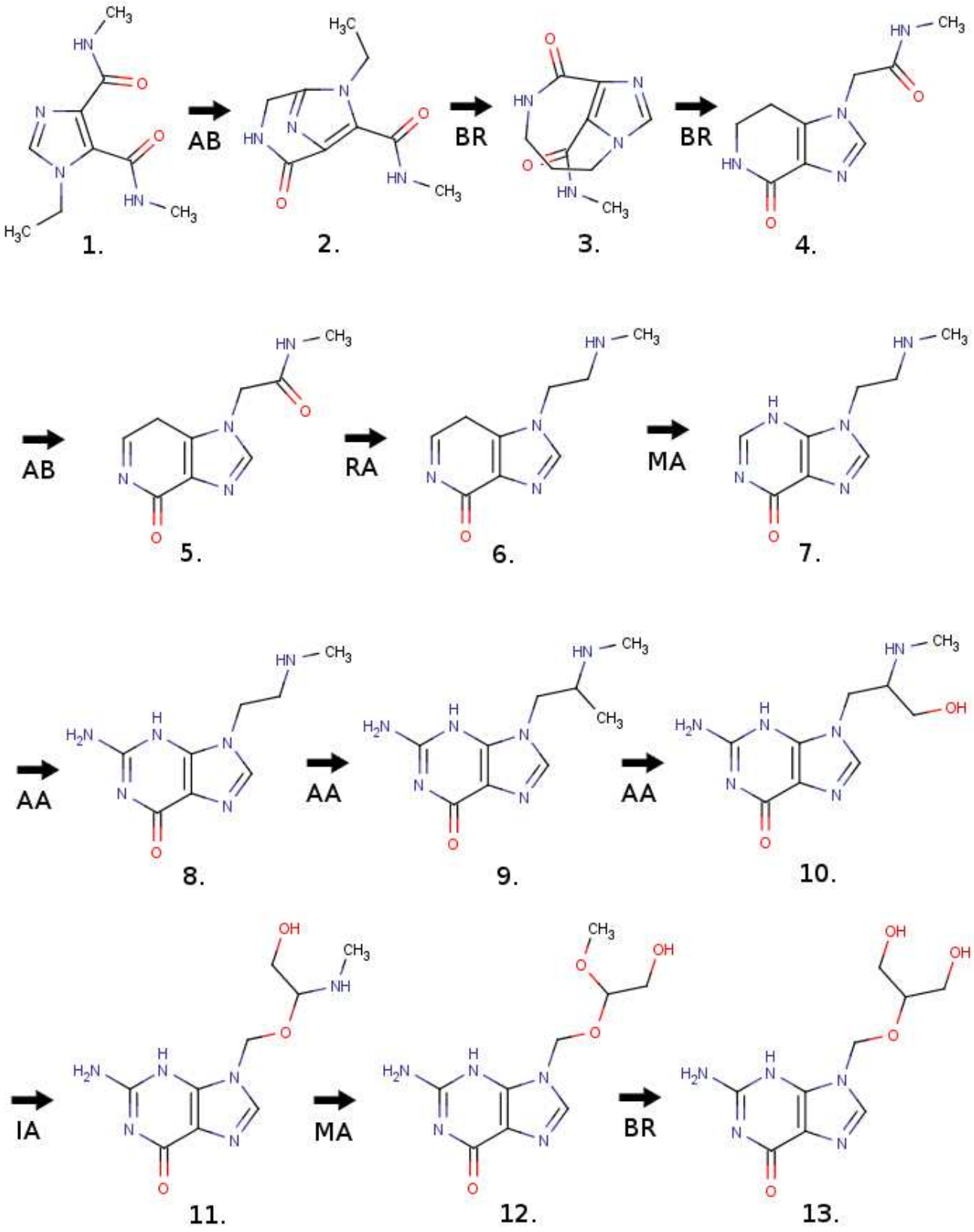
Zhodnocení

Z výše uvedených statistik je vidět, že technika scaffold hopping je ve srovnání s klasickým generováním cesty úspěšnější co do počtu scaffoldů. Nemusí vždy platit, že výsledné cesty s technikou scaffold hopping obsahují více různých scaffoldů než ty bez ní – to platí hlavně pro krátké cesty. Často jsou cesty stejně rozmanité a v minimálním počtu případů cesta se zapnutou technikou scaffold hopping bývá strukturně méně zajímavá než ta s vypnutou technikou. Pravděpodobnější ale je

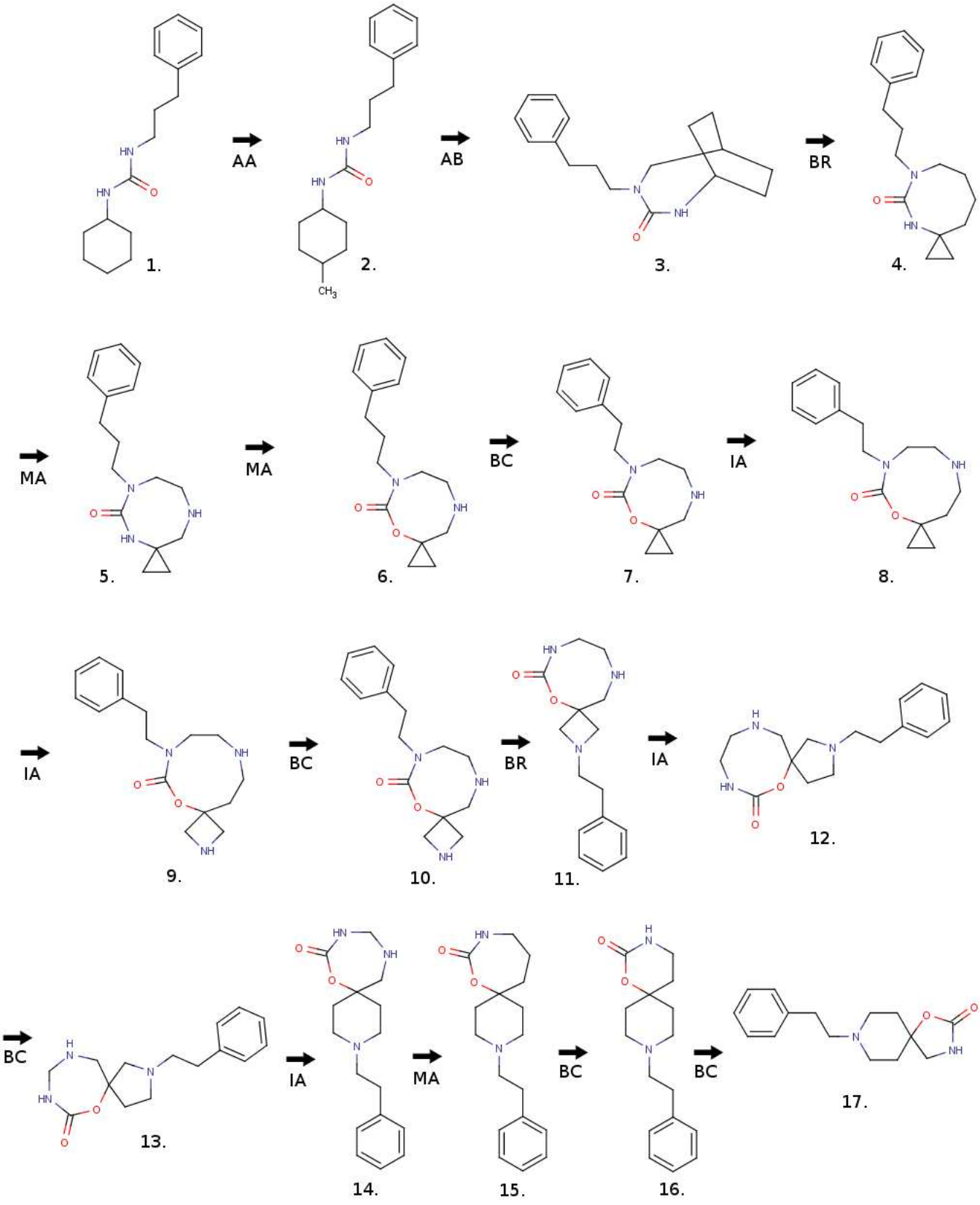
ta skutečnost, že s technikou scaffold hopping bude výsledná cesta rozmanitější než aby platilo, že s technikou bude méně rozmanitá. Podle očekávání jsou tedy obecně množiny výsledných molekul s technikou scaffold hopping strukturně rozmanitější, což by mohlo být užitečné v drug discovery procesu.



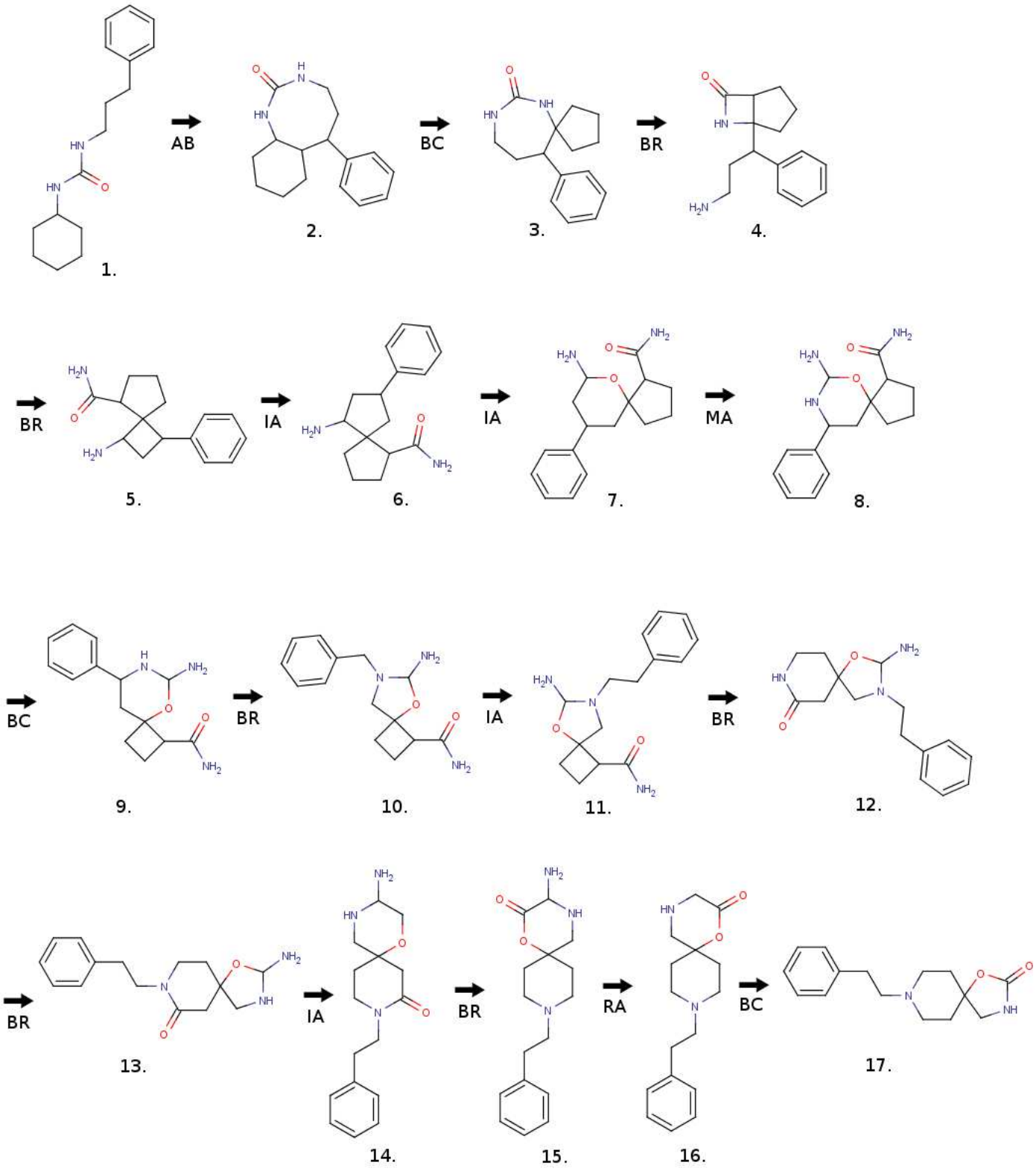
Obrázek 3.20: Cesta při vypnuté technice scaffold hopping.



Obrázek 3.21: Cesta při zapnuté technice scaffold hopping.



Obrázek 3.22: Cesta při vypnuté technice scaffold hopping.



Obrázek 3.23: Cesta při zapnuté technice scaffold hopping.

Závěr

Výsledkem této diplomové práce je, že nově přidaný další způsob generování cesty je funkční a integrovaný do aplikace Molpher.

Původní myšlenka modifikace spočívala v tom, že se na každé scaffold úrovni budou iterativně generovat cesty mezi každou dvojicí již nalezených sousedních molekul na cestě. Tento přístup byl z logických důvodů změněn a je popsán v textu práce. Konečná verze provádění techniky scaffold hopping spočívá v generování cesty mezi předposlední molekulou z aktuálně známé cesty a cílovou. Nicméně princip zůstal stejný. Po skončení každé scaffold úrovně obsahuje daná cesta různé scaffoldy. Tedy různorodé sloučeniny.

Z experimentů vyplynulo, že s technikou scaffold hopping kvalita nalezených cest zůstala stejná. Časově je aplikace pomalejší (plyne z principu), avšak dlouhé cesty nalézá rychleji a výsledné molekuly obsahují více různých scaffoldů, což je ceněná vlastnost v drug discovery procesu.

Vytvořená aplikace, kterou jsem použil pro testování kvality cesty (viz odstavec 3.2), by mohla beze změn sloužit i nadále ke zjišťování kvality nalezených cest před libovolnou modifikací a po ní. Nic tomu nebrání, jelikož obecně zjišťuje vzdálenosti generovaných cest od zadané množiny sloučenin.

Jako možnou budoucí práci mohu zmínit rozšíření techniky scaffold hopping o uživatelsky nastavitelnou množinu úrovní, kde se budou cesty generovat. Uživatel by si sám mohl zvolit, které z existujících čtyř úrovní budou použity. Další možnost rozšíření vidím v přidání jiné scaffold úrovně (se zachováním posloupnosti úrovní od nejobecnější k nejpřesnější).

Seznam použité literatury

[1] Bemis, G. W. a Murcko, M. A. "The properties of known drugs. 1. Molecular frameworks". Journal of Medicinal Chemistry. 1996, 39(15), 2887 – 2893. Dostupné z http://pubs.acs.org/doi/abs/10.1021/jm9602928.

[2] Böhm, H. J.; Flohr, A. a Stahl, M. "Scaffold hopping". Drug Discovery Today: Technologies. 2004, 1(3), 217 – 224. Dostupné z http://www.sciencedirect.com/science/article/pii/S1740674904000460.

[3] Boost. [online]. Boost.org. Dostupné z http://www.boost.org

[4] Chemical Abstracts. [online]. Chemical Abstracts Service. Dostupné z http://www.cas.org

[5] Colliandre, L.; Guilloux, V. L.; Bourg, S. a Morin-Allory, L. "Visual characterization and diversity quantification of chemical libraries: 2. Analysis and selection of size-independent, subspace-specific diversity indices". Journal of Chemical Information and Modeling. 2012, 52(2), 327 – 342. Dostupné z http://pubs.acs.org/doi/abs/10.1021/ci200535y.

[6] CORBA. [online]. Object Management Group. Dostupné z http://www.omg.org/spec/CORBA/

[7] Drug discovery. [online]. Poslední aktualizace 2014 [cit. 2014-07-10]. Dostupné z http://www.nature.com/subjects/drug-discovery.

[8] Estrogen. [online]. Poslední aktualizace 4. července 2014 18:23 [cit. 2014-07-18]. Wikipedie. Dostupné z http://en.wikipedia.org/wiki/Estrogen.

[9] Geppert, H.; Vogt, M. a Bajorath, J. "Current trends in ligand-based virtual screening: molecular representations, data mining methods, new application areas, and performance evaluation". Journal of Chemical Information and Modeling. 2010, 50(2), 205 – 216. Dostupné z http://pubs.acs.org/doi/abs/10.1021/ci900419k.

[10] Guilloux, V. L.; Colliandre, L.; Bourg, S.; Guenegou, G.; Dubois-Chevalier, J. a Morin-Allory, L. "Visual characterization and diversity quantification of chemical libraries: 1. creation of delimited reference chemical subspaces". Journal of Chemical Information and Modeling. 2011, 51(8), 1762 – 1774. Dostupné z http://pubs.acs.org/doi/abs/10.1021/ci200051r.

[11] Hoksza, D.; Svozil, D. "Exploration of Chemical Space by Molecular Morphing". Bioinformatics and Bioengineering (BIBE). 2011 11th IEEE International Conference. Taiwan. ISBN 978-0-7695-4391-8. s. 201 – 208

[12] Indigo. [online]. GGA Software Services. Dostupné z http://www.ggasoftware.com/opensource/indigo

[13] Jolliffe, I. T. "Principal Component Analysis". Heidleberg, Germany: Springer. 2010.

[14] Hoksza, D.; Škoda, P.; Voršilák, M. a Svozil, D. "Molpher: a software framework for systematic chemical space exploration". Journal of Cheminformatics. 2014, 6(7). doi:10.1186/1758-2946-6-7. Dostupné z http://www.jcheminf.com/content/6/1/7.

[15] Kamada, T. a Kawai, S. "An algorithm for drawing general undirected graphs". Information Processing Letters (Elsevier). 1989, 31(1), 7 – 15. Dostupné z http://pubs.acs.org/doi/abs/10.1021/ci600338x.

[16] Kirkpatrick P.; Ellis C. "Chemical space". Nature. [online]. 2004, 432(7019). [cit.2014-02-16]. Dostupné z http://www.nature.com/nature/insights/7019.html.

[17] Koch, M. A.; Schuffenhauer, A.; Scheck, M.; Wetzel, S.; Casaulta, M.; Odermatt, A.; Ertl, P. a Waldmann, H. "Charting biologically relevant chemical space: a structural classification of natural products (SCONP)". USA: The National Academy of Sciences. 2005, 102(48), 17272 – 17277. Dostupné z http://www.pnas.org/content/102/48/17272.long.

[18] SW projekt Molpher. [online]. 2012 [cit. 2014-03-18]. Dostupné z https://trac.assembla.com/molpher/wiki.

[19] MPICH. [online]. Dostupné z http://www.mpich.org

[20] Open Babel. [online]. Open Babel development team. Dostupné z http://openbabel.sourceforge.net

[21] OpenMP. [online]. OpenMP®. Dostupné z http://openmp.org/wp

[22] Open MPI. [online]. Dostupné z http://www.open-mpi.org

[23] PubChem Compound Database. [online]. The PubChem Project. Dostupné z http://www.ncbi.nlm.nih.gov/pccompound?term=all[filt]&cmd=search

[24] Oprea, T. I. a Gottfries, J. "Chemography: the art of navigating in chemical space". J. Comb. Chem.. 2001, 3(2), 157 – 166. Dostupné z http://pubs.acs.org/doi/abs/10.1021/cc0000388.

[25] Qt. [online]. Qt Project. Dostupné z http://qt-project.org/doc/

[26] RCF. [online]. Delta V Software. Dostupné z http://www.deltavsoft.com/RCF.html

[27] Receptor (biochemistry). [online]. Poslední aktualizace 10. července 2014 03:23 [cit. 2014-07-18]. Wikipedie. Dostupné z http://en.wikipedia.org/wiki/Receptor_%28biochemistry%29.

[28] Renner, S.; van Otterlo, W. A.; Dominguez Seoane, M.; Mocklinghoff, S.; Hofmann, B.; Wetzel, S.; Schuffenhauer, A.; Ertl, P.; Oprea, T. I.; Steinhilber, D.; Brunsveld, D.; Rauh, D. a Waldmann, H. "Bioactivity-guided mapping and navigation of chemical space". Nature Chemical Biology. 2009, 5(8), 585 – 592. Dostupné z http://www.nature.com/nchembio/journal/v5/n8/full/nchembio.188.html.

[29] RDKIT. [online]. Greg Landrum. RDKit: Open-source cheminformatics. Dostupné z http://www.rdkit.org

[30] SCHIFFMAN, S.; REYNOLDS, M. L. a YOUNG, F. W. "Introduction to Multidimensional Scaling: Theory, Methods, and Applications". Bingley, United Kingdom: Emerald Group Publishing Limited. 1981.

[31] SCHUFFENHAUER, A.; ERTL, P.; ROGGO, S.; WETZEL, S.; KOCH, M. A. a WALDMANN, H. "The scaffold tree–visualization of the scaffold universe by hierarchical scaffold classification". Journal of Chemical Information and Modeling. 2007, 47(1), 47 – 58. Dostupné z http://pubs.acs.org/doi/abs/10.1021/ci600338x.

[32] SMILES. [online]. Poslední aktualizace 16. května 2014 03:02 [cit. 2014-07-20]. Wikipedie. Dostupné z http://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system

[33] STRATEGY. [online]. Dostupné z http://ulita.ms.mff.cuni.cz/pub/predn/NPRG024/21

[34] SILICOS-IT. Strip-it^{TM}. [software]. [přístup 2013]. Dostupné z http://silicos-it.com/software/strip-it/1.0.2/strip-it.html.

[35] SUN, H.; TAWA, G. a WALLQVIST, A. "Classification of scaffold-hopping approaches". Drug Discovery Today. 2012, 1(7-8), 310 – 324. Dostupné z http://www.sciencedirect.com/science/article/pii/S1359644611003813.

[36] THREADING BUILDING BLOCKS. [online]. Intel. Dostupné z http://www.threadingbuildingblocks.org

[37] TODESCHINI, R. a CONSONNI, V. "Handbook of Molecular Descriptors". Weinheim, Germany: Wiley-VCH. 2000, 11, 1 – 668. Dostupné z http://onlinelibrary.wiley.com/doi/10.1002/minf.v30.1/issuetoc.

[38] TVERSKY INDEX. [online]. Dostupné z http://www.daylight.com/dayhtml/doc/theory/

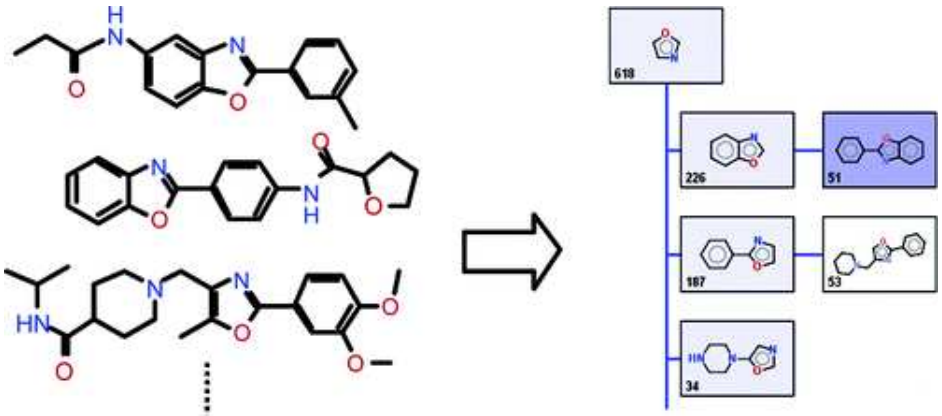
[39] VARNEK, A. a BASKIN, I. I. "Chemoinformatics as a theoretical chemistry discipline". Molecular Informatics. 2011, 30(1), 20 – 32. Dostupné z http://onlinelibrary.wiley.com/doi/10.1002/minf.v30.1/issuetoc.

[40] WETZEL, S.; KLEIN, K.; RENNER, S.; RAUH, D.; OPREA, T. I.; MUTZEL, P. a WALDMANN, H. "Interactive exploration of chemical space with Scaffold Hunter". Nature Chemical Biology. 2009, 5(8), 581 – 583. Dostupné z http://www.nature.com/nchembio/journal/v5/n8/full/nchembio.187.html.

Seznam použitých zkratek

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CID	Compound Identification Number
CORBA	Common Object Request Broker Architecture
DRCS	Delimited Reference Chemical Subspace
GPL	General Public License
GUI	Graphical User Interface
IA32	Intel Architecture, 32-bit
MDS	Multidimensional Scaling
MPI	Message Passing Interface
MPICH	MPI over CHameleon
OpenMP	Open Multi-Processing
PCA	Principal Component Analysis
RCF	Remote Call Framework
SDF	Structure-Data File
SIMD	Single Instruction, Multiple Data
SMILES	Simplified Molecular-Input Line-Entry System
SSE	Streaming SIMD Extensions
STL	Standard Template Library
TBB	Threading Building Blocks
TLS	Thread-local storage

Přílohy



Obrázek 3.24: Strom scaffoldů.
Použito z `http://pubs.acs.org/doi/abs/10.1021/ci600338x`.