ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Genetické algoritmy pro generování molekul
Student:	Jonatan Matějka
Vedoucí:	Mgr. Jan Starý, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

1. Vyberte či navrhněte způsob formálního popis vazeb mezi fragmenty molekul.
   Využijte existující databáze takových fragmentů a jejich formáty.
2. Implementujte genetické algoritmy, které z těchto fragmentů budou generovat molekuly,
   a testujte jejich fotochemické vlastnosti, především absorpci a emisi světla daných vlnových délek.
3. K chemickým výpočtům využijte software pro virtuální screening molekul vytvořený v BP Štěpána Sršně.
4. Implementaci proveďte v jazyce C. Dbejte na korektnost, čitelnost, rozšiřitelnost a přenositelnost kódu.
5. Vytvořte uživatelskou a programátorskou dokumentaci, nejlépe ve formě standardní manuálové stránky.
6. Aplikaci řádně otestujte.

Seznam odborné literatury

[1] Štěpán Sršeň: Distribuovaná infrastruktura pro virtuální screening molekul. Bakalářská práce FIT ČVUT (2017).
Yinan Shu, Benjamin Levine: Simulated evolution of fluorophores for light emitting diodes, Journal of Chemical Physics 142 (2015).

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 28. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Genetické algoritmy pro generování molekul

Jonatan Matějka

Vedoucí práce: Mgr. Jan Starý, Ph.D.

12. května 2017

Poděkování

Za poctivý dohled, opravdový zájem, cenné rady a věnovaný čas děkuji vedoucímu práce Mgr. Janu Starému, Ph.D. Dále děkuji správcům laboratoře SAGElab za poskytnuté výpočetní prostředky.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2017 .....

Odkaz na tuto práci

Matějka, Jonatan. *Genetické algoritmy pro generování molekul*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem této práce je vytvoření programu, který bude na základě požadavku uživatele generovat odpovídající molekuly. Program za pomoci genetického algoritmu vytváří z uživatelem dodané databáze populace nových molekul, které jsou následně ohodnoceny virtuálním screeningem – softwarem pro určení vhodnosti dané molekuly podle uživatelského zadání. Byla vybrána vhodná reprezentace molekuly a navrženy příslušné genetické operátory křížení a mutace. Výsledkem je sada unixových programů provádějící jednotlivé úkony genetického algoritmu. Výsledný program umožňuje ze zadání pro virtuální screening a vstupní databáze molekul vygenerovat rozsáhlou sadu molekul odpovídající zadaným požadavkům.

Klíčová slova genetický algoritmus, evoluční algoritmus, generování molekul, virtuální screening, molekula, graf molekuly, OpenBabel

Abstract

The purpose of this work is to produce a program which generates suitable molecules based on user's request. Starting with a database of molecules provided by the user, generations of new molecules are generated and evaluated by a virtual molecule screener, obtaining a measure of fitness with respect to the request. We choose a suitable data model of a molecule and design appropriate genetic operators of crossover and mutation. The result is a set of unix utilities performing the individual steps of a genetic algorithm. Based on the user's input, the program produces a large set of molecules satisfying the given constraints.

Keywords genetic algorithm, evolutionary algorithm, molecule generation, virtual screening, molecule, molecular graph, OpenBabel

Obsah

Seznam obrázků

Úvod

Při vývoji nových léčiv nebo materiálů požadujeme od hledaných sloučenin jisté vlastnosti. Zkoumání daných vlastností v laboratoři je ale velice nákladné a zdlouhavé. Proto je laboratorní práce doplněna o virtuální screening. Jedná se o soubor výpočtů, který umožňuje pro daná kritéria s určitou přesností rozhodnout, o jak vhodnou molekulu jde. Na základě tohoto výsledku pak můžeme výrazně zredukovat výběr možných sloučenin vyřazením nevhodných možností.

Vstupem do celého tohoto procesu jsou databáze molekul, například seznamy komerčně dostupných látek. Pokud ale počítáme s následnou syntézou nalezených látek, můžeme vstupní databázi generovat. Neomezujeme se tak nabídkou trhu, naopak prozkoumáváme všechny varianty, bez ohledu na náročnost jejich získání. Tradičním přístupem je pak generování molekul podle strategie odpovídající danému zadání. Výsledkem jsou specializované generátory pro každý typ problému.

Tato práce se zaměřuje na generování za pomoci genetického algoritmu. Simulováním průběhu evoluce mezi generacemi molekul postupně dospíváme k vhodnějším řešením. Vhodnost každé molekuly je určena za pomoci virtuálního screeningu, který tak neslouží ke koncovému ohodnocení molekul, ale k neustálému posuzování nově generovaných molekul.

Kapitola 1

Cíl práce

Cílem práce je vytvořit program, který bude generovat molekuly vyhovující požadavkům uživatele. Generování bude probíhat pomocí genetického algoritmu a vhodnost jednotlivých molekul bude posuzována virtuálním screeningem. Při spouštění programu uživatel dodá prvotní populaci molekul a zadání pro virtuální screening. Opakovaným aplikováním genetických operátorů výběru, křížení, mutace a začlenění bude postupně docházet k nálezu lepších jedinců. S přibývajícími generacemi se populace nových molekul budou přibližovat k požadavkům uvedeným v zadání pro virtuální screening. Je nutné nalézt vhodnou reprezentaci molekuly pro genetický algoritmus a navrhnout operátory křížení a mutace. Pro ohodnocení molekul bude použit software vyvinutý kolegou Štěpánem Sršněm v rámci práce *Distribuovaná infrastruktura pro virtuální screening molekul*. Jednotlivé operátory budou implementovány v jazyce C a průběh genetického algoritmu bude řídit shellový skript. Program bude přenositelný mezi unixovými systémy a jednoduše rozšiřitelný o další funkcionalitu. Všechny části programu budou zdokumentovány ve standardních manuálových stránkách.

Kapitola 2

Genetické algoritmy

Koncept genetického algoritmu uvedl v 70. letech minulého století John H. Holland [1]. Inspirací je přírodní proces, kde se jedinci skrz generace díky evoluci postupně vyvíjejí do vhodnějších podob. Stejně jako evoluce hledá nejvhodnějšího jedince pro dané prostředí, genetický algoritmus hledá nejvhodnější řešení pro daný problém.

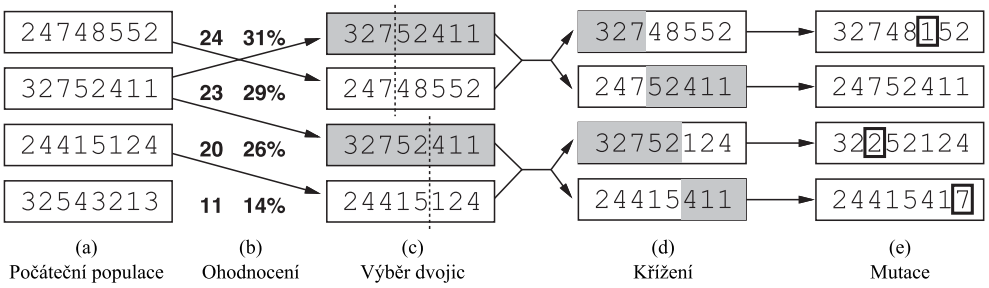
V původním návrhu je každé řešení reprezentováno *chromozomem* – binárním řetězcem pevné délky. Každý chromozom je možné interpretovat v kontextu našeho problému jako řešení a určit jeho vhodnost tzv. *fitness* nebo ohodnocovací funkcí. Soubor jedinců v rámci jedné *generace* se nazývá *populace*.

Jedinci, kteří dosáhli dobrého ohodnocení, podstupují *křížení* (rekombinaci). Vybrané dvojice chromozomů jsou zduplikovány, následně je vybrán bod, ve kterém dojde k rozdělení obou řetězců. Oddělené části jsou prohozeny a připojeny k řetězci opačného jedince. Tímto procesem vzniknou dva potomci. Operace výběru jedinců pro křížení se nazývá *selekce*.

Na některé z potomků je následně aplikována *mutace*. Z řetězce je vybrán jeden bit a jeho hodnota je invertována.

Tento proces křížení a mutací se opakuje než vytvoříme novou populaci složenou z potomků. Tito noví jedinci nahradí část nebo celou populaci předchůdců, dojde tak k *začlenění* a postupu do další generace.

Ne vždy je binární řetězec vhodnou reprezentací daného problému. Pro různé problémy můžeme použít například uspořádané n-tice celých čísel, vektory reálných čísel nebo stromy. Reprezentace problému v genetickém algoritmu se obecně nazývá *genotyp*. S každým dalším genotypem je nutné znovu navrhnout operátory křížení a selekce tak, aby dávaly smysl v kontextu dané reprezentace.



Obrázek 2.1: Osmice čísel z intervalu $\langle 1, 8\rangle$ reprezentující jednotlivá řešení problému osmi dam. Jedinci z počátení populace (a) jsou ohodnoceni fitness funkcí (b). Podle vhodnosti jedinců jsou vybrány dvojice (c), které postupují do křížení (d). Po provedení mutace (e) u několika jedinců dostáváme novou populaci potomků. [2]

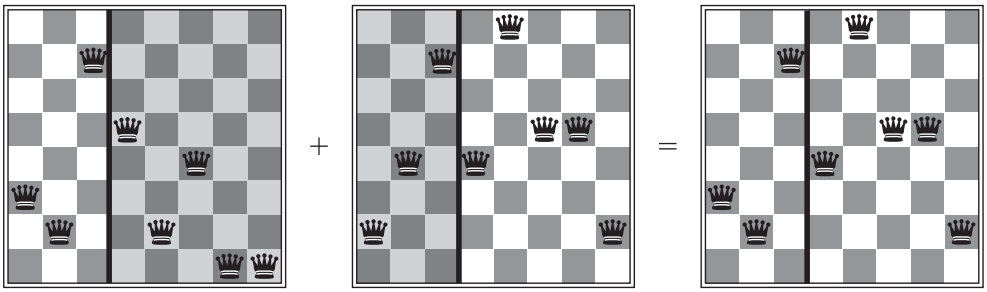
2.1 Problém osmi dam

Genetickým algoritmem je možné řešit mnoho různorodých problémů. Jednoduchým příkladem může být problém osmi dam – tradiční kombinatorická úloha spočívající v rozestavění osmi dam na šachovnici takovým způsobem, aby se žádné dvě z nich navzájem neohrožovaly. Neboli každý sloupec, řada i diagonála musí obsahovat nejvýše jednu dámu.

Pro účely genetického algoritmu stav šachovnice reprezentujeme jako osmici celých čísel z intervalu $\langle 1, 8\rangle$. Bereme v úvahu pouze takové stavy, kdy je v každém sloupci právě jedna dáma. Číslo v n-tém prvku osmice pak určuje řádek, ve kterém se nachází dáma z n-tého sloupce. Ohodnocovací funkce spočítá, kolik pro daný stav šachovnice existuje dvojic dam, které se navzájem neohrožují. Vrací tedy celé číslo z rozsahu $\langle 0, 28\rangle$, stavy ohodnocené číslem 28 jsou hledaným řešením problému.

Operátor křížení se nikterak neliší od původního návrhu popsaného výše. Pro dvě osmice vybereme jeden ze sedmi možných bodů, v něm provedeme rozdělení, prohození a spojení do nových osmic. Operátor mutace, tak jak byl původně navržen, dává smysl pouze nad bitovými řetězci. Pro naše osmice použijeme jiný návrh. Namísto přehození jednoho bitu nastavíme hodnotu jednoho prvku na náhodné celé číslo z intervalu $\langle 1, 8\rangle$.

Průběh algoritmu v rámci jedné generace je ilustrován obrázkem 2.1, význam křížení na konkrétní stavy šachovnice vidíme na obrázku 2.2.



Obrázek 2.2: Interpretace křížení prvních dvou jedinců z obrázku 2.1 v kontextu problému osmi dam. První potomek je výsledkem spojení levé části šachovnice prvního předchůdce a pravé části druhého předchůdce. [2]

2.2 Vyvinutá anténa

Genetické algoritmy nejsou určeny pouze k řešení teoretických úloh. Existuje mnoho oblastí v reálném životě, kde počítačem vyvinutá řešení jsou srovnatelná nebo lepší než řešení navržená člověkem. Notoricky známým příkladem je anténa vyvinutá pro NASA[1] misi Space Technology 5 (dále ST5). Za pomoci superpočítače a genetických algoritmů byla v roce 2004 vyvinuta anténa splňující požadovaná kritéria lépe než člověkem navržené řešení. V roce 2006 byly v rámci mise ST5 na oběžnou dráhu vyneseny tři satelity vybavené touto anténou [3].

Výzkumný tým připravil dva genetické algoritmy, které se odlišovaly zvoleným genotypem. První algoritmus byl zaměřen na antény vyrobené z jednoho kusu drátu (obr. 2.3 vpravo dole). Zvolenou reprezentací byla čtveřice trojrozměrných vektorů. Každý vektor určoval bod v prostoru, kterým prochází drát antény. Výsledný tvar antény byl lomená čára začínající v počátku, procházející třemi body v prostoru a končící ve čtvrtém bodě.

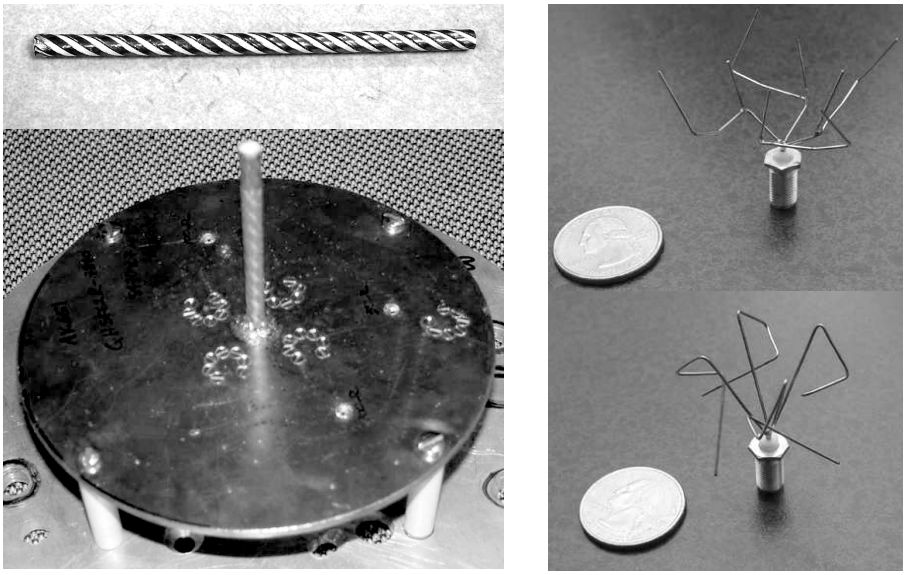
Druhý algoritmus byl zaměřen na antény z větvícího se drátu (obr. 2.3 vpravo nahoře). Reprezentací byl strom příkazů, vykonávaných nad drátem:

- `forward(length, radius)` – protažení drátu určité tloušťky do určité vzdálenosti v aktuální orientaci
- `rotate-x(angle)`, ... – změna orientace rotací kolem osy x, ...

Pro oddělení jednotlivých podstromů byly použity hranaté závorky. Reprezentace výsledné rozvětvené antény je vidět na obrázku 2.4.

Algoritmus v obou případech vyvinul jedno rameno antény. Výsledná anténa pak byla složena z původního ramene a tří jeho kopií otočených o 90°, 180° a 270°. [4]

[1] National Aeronautics and Space Administration.



Obrázek 2.3: Antény pro misi ST5. Vlevo: člověkem navržená, konvenčními postupy získaná QFH* anténa. Vpravo: počítačem vypočítané, genetickými algoritmy vyvinuté antény. [4]

*Quadrifilar helical antenna.

```
rotate-z(1.984442) 1 [rotate-x(2.251165) 1 [rotate-x(0.062240) 1
[rotate-x(0.083665) 1 [rotate-y(-2.449035) 1 [ rotate-z(-0.894357)
1 [rotate-y(-2.057702) 1 [rotate-y(0.661755) 1 [rotate-x(0.740703)
1 [rotate-y(2.057436) 1 [ forward(0.013292,0.000283) 2
[rotate-z(-1.796822) 1 [ rotate-x(-1.651348) 1 [rotate-y(-2.940880) 1
[rotate-x(0.095209) 1 [rotate-z(1.248723) 1 [forward(0.003815,0.000363)
1 [ forward(0.008289,0.000355) 1 [forward(0.008413,0.000369) 1 [
rotate-x(-0.006494) 1 [rotate-x(-0.592854) 1 [rotate-z(-2.085023) 1
[rotate-z(1.735374) 1 [rotate-z(-2.045125) 1 [ rotate-z(0.203076) 1
[rotate-z(1.750799) 1 [rotate-z(-2.038688) 1 [rotate-z(1.725007) 1
[rotate-y(1.478109) 1 [rotate-x(2.477117) 1 [rotate-x(-2.441858) 1
[forward(0.015082,0.000223) ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ]
rotate-y(2.335438) 1 [ rotate-y(-1.042201) 1 [rotate-y(-1.761594) 1
[rotate-x(2.518405) 1 [rotate-z(-0.739608) 1 [rotate-x(0.426553) 1 [
rotate-z(-0.291483) 1 [rotate-x(2.152738) 1 [ forward(0.013190,0.000414)
] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ]
```

Obrázek 2.4: Reprezentace vyvinuté rozvětvené antény z obrázku 2.3 vpravo nahoře. [4]

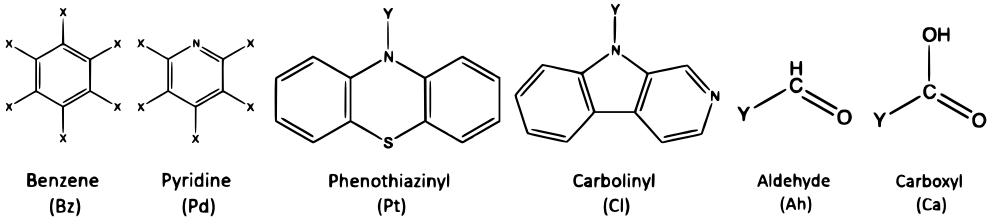
Kapitola 3

Analýza současných řešení

Použití genetického algoritmu v oblasti výpočetní chemie není žádnou novinkou. Široká využití nalezl při vývoji nových léčiv. Programy `AutoGrow` [5] nebo `OpenGrowth` [6] za pomoci genetického algoritmu staví *ligandy*, molekuly určené pro navázání na určitý receptor. Ohodnocení jednotlivých molekul má na starosti externí software, který provádí *docking*, neboli ověření vhodnosti ligandu pro konkrétní receptor. Variantou také může být vlastní, odlehčená, implementace dockingu, kdy jsou molekuly hodnoceny podle podobnosti vybraných částí oproti molekule, která je známým vhodným ligandem pro určitý receptor. Tak je tomu v případě programů `PRO_LIGAND` [7] nebo `SPROUT` [8].

Další využití genetického algoritmu mimo farmaceutický průmysl jsou v materiálovém a chemickém inženýrství. Příkladem může být program pro vývoj organických LED diod [9], jakožto levných světelných zdrojů pro displeje a další. Součástí programu je fitness funkce, která hodnotí vyvinuté molekuly, v tomto případě dle určité formy jejich svítivosti. Podobným případem je program pro hledání vhodných materiálů pro organické fotovoltaické články [10]. Fitness funkce pak naopak hodnotí schopnost pohlcovat světlo. Tématu světla ve fyzice molekul a virtuálnímu screeningu se do hloubky věnoval kolega Štěpán Sršeň ve své bakalářské práci [11].

V prozkoumaných pracích se objevovaly různé genotypy (způsoby zakódování molekul pro účely genetického algoritmu) – od jednoduchých reprezentací, kdy konečný vektor umožňoval omezené množství kombinací určitých *fragmentů* (částí molekul) [10], přes stromy fragmentů [9] až po komplexní reprezentace, kde genotypem byla samotná molekula uložená v konkrétním chemickém formátu [5]. V následující kapitole rozeberu dva konkrétní genotypy, jakým způsobem kódují molekulu a jaké z toho plynou výhody a nevýhody.



Obrázek 3.1: Výběr z databáze fragmentů, první dvě části zleva jsou určeny k použití v kořeni stromu, ostatní slouží jako listy. [9]

3.1 Strom jako genotyp

Autoři článku *Simulated evolution of fluorophores for light emitting diodes* [9] se rozhodli reprezentovat jednotlivé molekuly stromem fragmentů. V kontextu článku je fragment část molekuly obohacená o informaci, kde a jakým způsobem se daná část může připojovat k ostatním částem a tvořit tak větší celek. Uzly stromu odpovídají jednotlivým fragmentům a hrany mezi uzly pak odpovídají spojení jednotlivých částí do výsledné molekuly. Uživatelským vstupem do programu je databáze fragmentů a sada pravidel, jakými způsoby se části molekul mohou spojovat.

Pro účely článku byla vybrána databáze šestnácti molekul (výběr viz obr. 3.1) s následujícími pravidly:

- Pouze vybrané molekuly mohou sloužit jako kořen stromu.
- Fragmenty obsahují vazebná místa (zástupné atomy) označená X a Y.
- Vazebné místo typu X je možné propojit pouze s vazebným místem typu Y.

Databáze obsahovala šest fragmentů, určených pro uložení do kořene stromu, o čtyřech až šesti vazebných místech typu X. Ostatní molekuly měly každá po jednom vazebném místu typu Y. Tudíž všechny fragmenty, které nebyly určené jako kořeny stromů, sloužily jako listy. Výsledné stromy tak dosahovaly vždy hloubky 1.

3.1.1 Výhody

Jednoznačnou výhodou použití stromu je jednoduchost operátorů křížení a mutace. Genetické programování, aneb vývoj programu za pomoci genetického algoritmu, využívá stromové reprezentace, kde uzly obsahují operace, listy obsahují proměnné nebo hodnoty a celý strom tak tvoří výraz [12]. Díky rozsáhlým pracím v této oblasti jsou velice dobře zmapovány operátory křížení a mutace. Tyto poznatky je možné uplatnit i v obecnějších genetických algoritmech, které využívají stromy jako genotyp.

Křížení dvou jedinců probíhá následovně:

1. V obou stromech je náhodně vybrán podstrom.
2. Podstromy jsou vyjmuty a vyměněny s opačným jedincem.
3. Vyměněné podstromy jsou připojeny na místo původního podstromu.

Mutace je výrazně rozmanitější, existuje mnoho různých implementací [13], např:

- Nahrazení vybraného podstromu náhodně vygenerovaným stromem.
- Nahrazení vybraného vnitřního uzlu vybraným podstromem.
- Prohození potomků vybraného vnitřního uzlu (v případě uspořádaného stromu).
- Prohození dvou vybraných disjunktních podstromů.

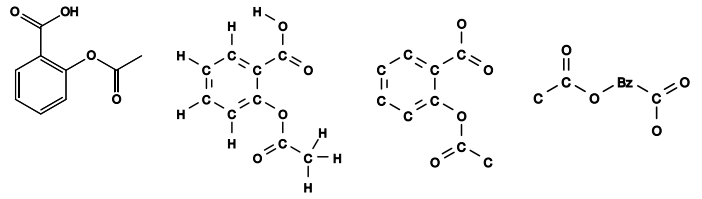
V článku byla mutace definována následovně:

1. Ze všech uzlů všech stromů se vybere náhodná podmnožina, nad kterou se bude provádět mutace.
2. Každý vybraný uzel je nahrazen takovým náhodně vybraným fragmentem, který má vazebné místo potřebné k propojení s rodičem daného uzlu.
3. Pokud má daný uzel nějaké potomky, jsou zachováni s ohledem na počet a typy vazebných míst nově zvoleného fragmentu. V případě nedostatku vazebných míst jsou přebyteční potomci odstraněni.
4. V případě přebytku vazebných míst je pro každé volné místo vybrán a připojen náhodný, s vazebným místem kompatibilní, fragment.
5. Předchozí krok se rekurzivně opakuje pro všechny takto nově vzniklé potomky, dokud se nedosáhne stromu, ve kterém jsou všechna vazebná místa zaplněna.

3.1.2 Nevýhody

Volba konkrétní databáze fragmentů a pravidel pro jejich vázání byla, jak autoři sami uvádějí, značně restriktivní a umožňovala konstrukci konečné a velice malé množiny molekul[2]. Zvolením jiné databáze a pravidel by bylo možné generovat hlubší stromy i nekonečné množiny molekul. Program ale obsahuje omezení, které nelze obejít vhodnou volbou uživatelského vstupu. Každé dva

[2]Celkem 1.26×10^6 možných molekul. [9]



	C	C	C	O	O	O	O	Bz
C	0	0	0	2	1	0	0	1
C	0	0	1	0	0	1	2	0
C	0	1	0	0	0	0	0	0
O	2	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	0
O	0	1	0	0	0	0	0	1
O	0	2	0	0	0	0	0	0
Bz	1	0	0	0	0	1	0	0

Obrázek 3.2: Zleva doprava: Strukturní vzorec Aspirinu, molekulární graf, zjednodušený molekulární graf, graf s fragmentem benzenového jádra, matice sousednosti.

fragmenty je možné propojit pouze jednou vazbou. Není tak možné vytvořit nový cyklus pouhým propojením dvou částí. Cyklus nelze vytvořit ani kruhovým propojením uzlů již z povahy stromu (absence kružnic je vlastnost vyplývající z definice stromu). Veškeré cykly v koncové molekule tak musí být obsaženy již v jednotlivých fragmentech. Jedná se spíše o teoretické než nějakým způsobem závažné omezení, takovéto řešení je zdaleka dostačující pro většinu použití.

Druhou nevýhodou je speciální formát fragmentů. Před použitím je nutné ručně upravit jednotlivé molekuly a specifikovat jejich vazebná místa. Náhradním řešením je vyvinout nástroj na automatizaci takového úkonu. Taktéž je potřeba připravit pravidla pro spojování fragmentů, aby odpovídala vstupní databázi.

3.2 Matice sousednosti jako genotyp

Naprosto odlišný způsob reprezentace molekuly zvolili autoři programu `EvoMD` [14]. Pracují se zjednodušenými molekulárními grafy[3], kde jednotlivé atomy molekuly odpovídají uzlům grafu a vazby mezi atomy odpovídají hranám mezi uzly; zjednodušení spočívá ve vynechání veškerých atomů vodíku. Při interpretaci takového grafu je potřeba vodíky doplnit na místa, kde se očekává jejich přítomnost. Další specialitou programu je kombinování fragmentů s atomy – vybrané části molekul lze nahradit přiřazenou značkou (v případě článku je to benzenové jádro se značkou Bz).

Pro účel genetického algoritmu je tento graf zakódován jako matice sousednosti – čtvercová matice typu $N \times N$, kde N je počet uzlů grafu (atomů a fragmentů) a prvek v i-tém řádku a j-tém sloupci obsahuje typ hrany – 0 pro neexistující vazbu, 1 pro jednoduchou vazbu, 2 pro dvojnou, atd. Souvislost mezi molekulou, jejím grafem a maticí sousednosti je vidět na obrázku 3.2.

[3]Hydrogen-depleted (hydrogen-suppressed) molecular graph. [15]

3.2.1 Výhody

Nespornou výhodou matice sousednosti je schopnost vyjádřit veškeré možné molekulární grafy. Navíc pro tento genotyp autoři vymysleli operátory křížení a mutace, které jsou lehké na implementaci. Křížení matic je inspirováno klasickým postupem z genetického algoritmu:

1. V obou jedincích vybraných pro křížení je zvolen bod na diagonále. Pro matici typu $N \times N$ je voleno celé číslo x z intervalu $\langle 1, N)$.

2. Matice je rozdělena na čtyři podmatice podle x-tého řádku a sloupce. Pravá horní a levá dolní matice je zahozena. Zbydou tak dvě čtvercové matice o velikostech x a $N - x$.

3. Podmatice vzniklé z pravých dolních částí jsou prohozeny s opačným jedincem.

4. Podmatice jsou slepeny zpět do celé čtvercové matice, chybějící pravá horní a levá dolní část jsou vyplněny nulami.

5. Oddělené komponenty grafu jsou spojeny náhodně vybranými hranami do souvislého grafu.

6. Je vybrán takový náhodný cílový počet hran E, který splňuje nerovnici $N - 1 \leq E \leq \binom{N}{2}$

7. Do matice jsou doplněny náhodné hrany tak, aby jejich počet nebyl nižší než vybraný cíl E.

Výsledkem jsou dva noví jedinci. Samotnému křížení ještě předchází náhodná permutace řádků (a příslušných sloupců) matice, aby docházelo k rovnoměrnému výběru všech sloupců. Během křížení je potřeba náležitě udržovat a vyměňovat informace o typu uzlu, které se vážou ke každému řádku a sloupci (na obrázku 3.2 znázorněno v záhlaví tabulky).

Popis fungování mutace byl v článku rozdělen na tři podoperátory:

- Změna počtu uzlů grafu odebráním náhodného uzlu nebo přidáním náhodného atomu či fragmentu. Hrany připojené k odstraňovanému uzlu jsou přepojeny k náhodně vybraným uzlům. K nově přidanému uzlu jsou připojeny hrany od náhodně vybraných uzlů.

- Změna počtu hran grafu přidáním nebo odebráním náhodné hrany. Po odebrání hrany je opravena souvislost grafu, stejně jako při křížení.

- Výměna dvou typů uzlů. Dvěma náhodně vybraným uzlům je prohozena informace, kterému atomu nebo fragmentu náleží.

3.2.2 Nevýhody

Očividnou nevýhodou výše popsaných operátorů je velice nevhodně navržené křížení. V jeho průběhu dochází ke ztrátě až poloviny informace obsažené v matici. Její místo je následně vyplněno opravným algoritmem (viz kroky křížení 5–7) na základě náhody. Při dělení matice také není zaručeno, že dojde k rozdělení na dvě souvislé části. Je naopak vysoce pravděpodobné, že oddělená podmatice obsahuje více nesouvislých komponent. Spojování komponent do souvislého grafu probíhá již zmíněným opravným algoritmem. Program se tak odchyluje od genetického algoritmu k náhodnému prohledávání.

Schopnost matice sousednosti vyjádřit veškeré možné molekulární grafy má i svoji odvrácenou stranu mince. Problémem celého programu je absence jakýchkoliv omezení stupně jednotlivých uzlů (vaznosti atomů nebo fragmentů). Vznikají tak i grafy molekul, které podle fyzikálních pravidel zcela zjevně nemohou existovat.

Pokud uživatel požaduje, aby výstupem byla trojrozměrná podoba molekuly, je nutné ji nejprve vygenerovat z daného grafu molekuly. Prostorové uspořádání atomů molekuly ale nemusí být jediné možné, nebo nemusí vůbec existovat.

Kapitola 4

Návrh

Program navržený pro tuto práci se chová následujícím způsobem: Uživatel dodá vstupní sadu molekul a zadání pro `vms` – program vytvořený v rámci práce *Distribuovaná infrastruktura pro virtuální screening molekul* [11]. Ze vstupních molekul je vytvořena prvotní populace a ohodnocena pomocí `vms`. Následně probíhají standardní kroky genetického algoritmu – výběr, křížení, mutace, ohodnocení nově vytvořených jedinců a začlenění do populace následující generace.

Každý z těchto kroků odpovídá samostatné aplikaci. Koordinaci činnosti jednotlivých částí genetického algoritmu má na starosti jeden centrální skript. Díky tomu je program jednoduše upravitelný a rozšiřitelný, ať už výměnou jednotlivých částí, nebo změnou běhu skriptu.

4.1 Graf jako genotyp

Klíčovou otázkou při návrhu genetického algoritmu je volba genotypu. Výběr byl zúžen několika faktory. `vms` vyžaduje vstup molekul ve formátu XYZ – výčet atomů molekuly v kartézských souřadnicích [16]. Pro neustálé ohodnocování jedinců je vhodné, aby jejich reprezentace byla v podobném nebo jednoduše převoditelném formátu. Prvním omezením tedy byla nutnost udržovat pozici jednotlivých atomů v prostoru. Pro operátory křížení a mutace je také vhodné, aby formát obsahoval informace o jednotlivých vazbách mezi atomy. Vzhledem k těmto omezením padla volba na formát SDF – obsahuje výčet jednotlivých atomů a jejich vazeb [17]. Ukázka uložení molekuly vody v tomto formátu je na obrázku 4.1. Kromě těchto příhodných vlastností je také velmi rozšířeným formátem a oblíbenou volbou pro ukládání obecných chemických databází [11].

Všechny části programu přijímají a produkují soubory ve formátu SDF. Zvoleným genotypem je tedy graf, kde každý uzel obsahuje informaci o typu atomu a jeho poloze a každá hrana obsahuje informaci o jejím typu (násobnosti).

```
962
  -OEChem-04291706313D

  3  2  0     0  0  0  0  0  0999 V2000
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.2774    0.8929    0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.6068   -0.2383   -0.7169 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
M  END
$$$$
```

Obrázek 4.1: Ukázka molekuly vody ve formátu SDF. [18]

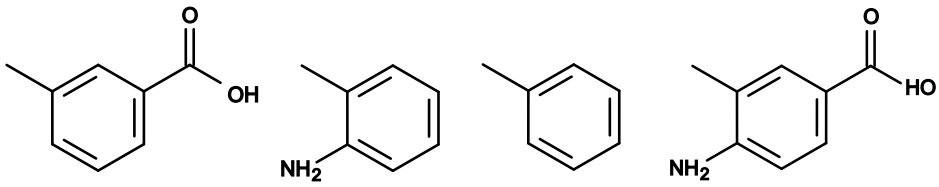
4.1.1 Výhody

Graf obohacený o výše zmíněné informace umožňuje navrhnout operátory křížení a mutace takovým způsobem, aby poskytovaly mnoho možností, jak molekuly mezi sebou kombinovat a měnit. Zároveň ale tyto operace dostatečně omezuje, aby každým kombinováním i úpravou vznikla vždy validní molekula, která by splňovala základní fyzikální pravidla pro její existenci.

Operátor křížení je podobný programu `LigMerge` [19], na kterém je založené křížení v programu `AutoGrow` [5]. Kombinace dvou molekulárních grafů programem `LigMerge` probíhá následovně:

1. Naleznou se dvojice všech společných (navzájem izomorfních) podgrafů.
2. Z nalezených dvojic jsou vyřazeny takové páry, které mají navzájem rozdílné geometrie (prostorové uspořádání atomů). Dvě geometrie jsou stejné právě tehdy, když existuje zobrazení jedné geometrie na druhou složené z posloupnosti posunutí a rotace.
3. Ze zbylých dvojic je vybrána jedna z největších. Velikostí dvojice je myšlen počet uzlů jednoho z podgrafů.
4. Grafy jsou posunuty a otočeny tak, aby se podgrafy z vybrané dvojice prostorově kryly.
5. Uzly a hrany, které se prostorově kryjí, jsou sloučeny. Tím dojde k sloučení původních grafů do jednoho výsledného.

Od výše popsaného algoritmu se operátor křížení liší v posledním bodě. Molekula vzniklá křížením může obsahovat pouze vybrané části z obou molekul. Společná část se stane jakýmsi jádrem, na které se připojí náhodně zvolené části (původně spojené s tímto jádrem) z obou molekul. Je toho dosaženo za pomoci takového hranového řezu společného jádra, který obě molekuly dělí na právě dvě komponenty. Nový jedinec je vytvořen z jedné komponenty první



Obrázek 4.2: Průběh algoritmu `LigMerge`. Zleva doprava: dvě vybrané molekuly, největší nalezený společný podgraf, výsledný sloučený graf.

molekuly, zbylé komponenty druhé molekuly a hran řezu propojující tyto dvě části.

Operátor mutace k nově vzniklé molekule připojí náhodně zvolený fragment, nebo ji náhodně ořízne. Připojovaný fragment je vybrán z uživatelem dodané databáze fragmentů, případně z prvotní populace. V případě ořezu se odstraní náhodné hrany a zachová se pouze největší zbylá komponenta.

4.1.2 Nevýhody

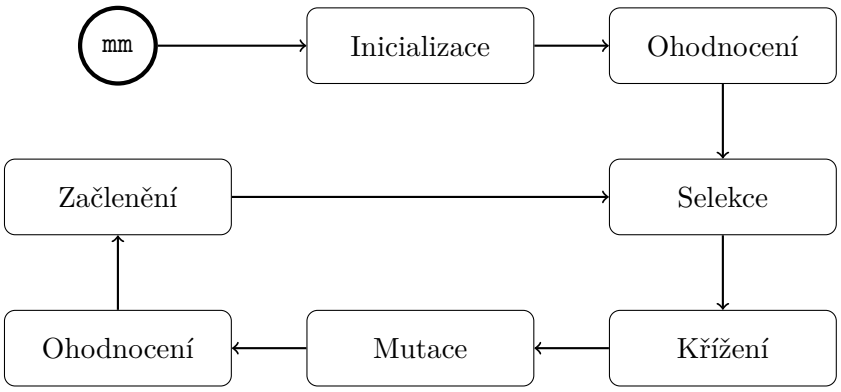
Hlavní nevýhodou je výpočetní náročnost a obtížná implementace výše zmíněných operátorů. Jednoduše popsané postupy, jako nalezení společných podgrafů, nebo posouzení rozdílnosti geometrie, jsou implementačně i výpočetně náročné. Vzhledem k očekávaným malým molekulám (nejčastěji do 100 atomů, maximálně pak do 1000 – formát SDF[4] je vnitřně omezený na nejvýše 999 atomů nebo vazeb) je doba výpočtu stále únosná a výpočetně nejdražší částí celého programu je ohodnocení molekul pomocí `vms`.

[4]Nejčastěji používaný SDF V2000 Molfile.

Kapitola 5

Implementace

Program pro tuto práci je navržen jako sada oddělených nástrojů úzce spolu spolupracujících. Nástroje `selection`, `crossover`, `mutation` a `inclusion` odpovídají příslušným operátorům genetického algoritmu a každému z nich je vyčleněna jedna podkapitola. Tyto aplikace jsou zastřešeny skriptem `mm`, který zajišťuje koordinaci jednotlivých částí. Program je navržen pro unixové systémy, tudíž jsou všechny nástroje implementovány v jazyce C a `mm` je unixový shellový skript. K překladu a instalaci je použita utilita `make`.



Obrázek 5.1: Zjednodušený diagram běhu programu `mm`.

5.1 Řízení genetického algoritmu

Srdcem celého projektu je skript `mm`. Volání z příkazové řádky vypadá následovně:

```
mm [-b backend] [-c cuts] [-e ratio] [-f frags] [-g steps]
    ↪ [-j jobs] [-m chance] [-n num] [-t size] [-x cache]
    ↪ request database outdir
```

Význam jednotlivých přepínačů a argumentů:

`-b backend` Použije program `backend` pro ohodnocování molekul. Výchozím programem je `vms`.[5]

`-c cuts` Maximální počet řezů provedených při křížení molekul (sekce 5.5).

`-e ratio` Poměr elity ku zbytku populace při začlenění (sekce 5.7).

`-f frags` Vstupní databáze fragmentů pro mutaci. Při neuvedení argumentu je použita `database`.

`-g steps` Počet evolučních kroků. Program se zastaví, když vypočítá `steps` generací. Výchozí volbou je 0, která značí běh programu do ukončení uživatelem.

`-j jobs` Počet paralelních běhů pro `backend`. Výchozí počet běhů je 4.

`-m chance` Pravděpodobnost provedení mutace jedné molekuly (sekce 5.6).

`-n num` Počet potomků vytvořených křížením (sekce 5.5).

`-t size` Velikost turnaje při selekci (sekce 5.4).

`-x cache` Soubor pro uložení vypočtených ohodnocení molekul (sekce 5.3). Výchozím souborem je `outdir/cache`.

`request` Vstupní požadavky pro `backend` v pro něj určeném formátu. Požadavky přijímané `vms` jsou popsány v práci *Distribuovaná infrastruktura pro virtuální screening molekul* [11].

`database` Vstupní databáze molekul pro vytvoření první populace. Formát databáze musí být jasný z přípony souboru (sekce 5.2).

`outdir` Výstupní složka pro průběžné ukládání výsledků genetického algoritmu.

5.2 Inicializace

V prvním kroku je ověřena validita uživatelem zadaných parametrů, přístupnost vstupních souborů a možnost zápisu do výstupního adresáře. Následně dojde ke konverzi vstupní databáze do formátu SDF pomocí nástroje `OpenBabel` [20]. Formát vstupního souboru je rozpoznán podle jeho přípony, je proto nutné ji vždy uvést. Databáze je použita jako prvotní populace a je uložena v souboru `outdir/generation.0`.

[5] Jako backend je možné použít libovolný program, který dodrží stejný formát vstupu a výstupu jako `vms`. Pro účely testování byl vytvořen program `fake_vms`, který má tyto požadované vlastnosti. Tato utilita byla použita při testovaní programu `mm` na systémech, kde nebylo možné nainstalovat `vms`.

5.3 Ohodnocení

V diagramu 5.1 se operace ohodnocení objevuje dvakrát. Nejprve dojde k ohodnocení prvotní populace. Následně jsou při každém evolučním kroku ohodnocovány nově vzniklé molekuly. `vms` vyžaduje vstupní formát XYZ. Konverze molekul je provedena za pomoci `OpenBabel`.

5.3.1 Fitness cache

Vyhodnocení fitness jedinců je velice drahou záležitostí. Ohodnocení jedné molekuly může trvat sekundy, minuty a v závislosti na vstupních požadavcích i hodiny. Je proto nezbytné, aby nedocházelo k opakovanému ohodnocování identických molekul a neplýtvalo se tak výpočetními prostředky. K těmto účelům byl vytvořen nástroj `eval_cache`. Pro jednoduchost porovnávání molekul se používá 27 znaků dlouhý molekulární hash InChIKey [21]. Hashe molekul jsou získány opět pomocí `OpenBabel`.

Program funguje ve třech módech:

```
eval_cache filter filter_hashes database hashes
eval_cache pre cachefile database hashes
eval_cache post cachefile hashes eval
```

V módu `filter` program projde databázi `database`, její hashe `hashes` a vyřadí z ní veškeré duplicity vzhledem k seznamu hashů `filter_hashes` i vůči databázi samotné. Program v tomto módu je použit před samotným ohodnocením k vyřazení jak duplicitních potomků, tak i potomků stejných jako některý z jedinců z předchozí populace. Zamezí se tak dvojímu ohodnocení stejné molekuly a také nedojde k zaplnění populace kopiemi jednoho úspěšného jedince. Díky použití tohoto filtru musí být každá molekula unikátní vůči své populaci.

Mód `pre` je použit po odfiltrování duplikátů a těsně před ohodnocením. Program projde databázi, hashe a porovná je vůči souboru s uloženými výsledky ohodnocení `cachefile`. Pokud nalezne uložený výsledek pro určitou molekulu, přesune ji na konec databáze. Výsledkem je databáze, kde se vrchní část skládá z nikdy neohodnocených molekul a zbytek jsou molekuly s již známým hodnocením. Přelom mezi těmito částmi, neboli počet molekul k ohodnocení je vypsán na standardní výstup programu. Tato informace je při hodnocení předána `vms` parametrem `-l` (pořadí poslední počítané molekuly v databázi [11]).

Po ohodnocení se použije mód `post`. Program zpracuje výsledky hodnocení `eval`, přiřadí k nim hashe příslušných molekul a uloží je do souboru `cachefile`. Zároveň výsledky doplní o dříve uložená hodnocení molekul, které se nacházely ve spodní části databáze.

Po dobu běhu programu `mm` se seznam uložených hodnocení postupně zvětšuje, jak dochází k ohodnocování jednotlivých populací. Tento soubor je možné

použít při dalším běhu programu pomocí argumentu `-x` za podmínky zachování vstupních požadavků a dále tak urychlit výpočet genetického algoritmu.

5.4 Selekce

Z ohodnocených jedinců je nutné vybrat páry, které se budou účastnit křížení. Výběr probíhá na základě turnajové selekce. Ze všech kandidátů je vybrána náhodná podmnožina o velikosti t. Jedinec s nejvyšší fitness z podmnožiny je vybrán do páru pro křížení. Velikost turnaje t je standardně 2 a lze ji nastavit parametrem `-t` při volání programu:

```
selection [-n num] [-t size] database eval output
```

Parametr `-n` určuje počet vybraných párů pro křížení. Pro vytvoření n potomků je potřeba $2n$ předchůdců, které spolu tvoří páry. Výchozí nastavení je `n=0`, které značí vytvoření přesně tolika potomků, kolik molekul obsahuje vstupní databáze. Program přečte databázi molekul `database`, jejich ohodnocení `eval` a vytvoří seznam párů `output`. Formát výstupního souboru je následující – na každém řádku jsou dvě čísla (odpovídající číslům molekul z databáze) tvořící jeden pár.

5.5 Křížení

Pro každý vybraný pár je provedeno křížení. Operátor se spouští následovně:

```
crossover [-c cuts] input selection output
```

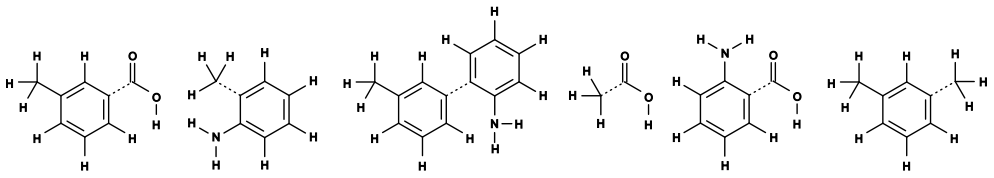
Program přečte databázi molekul `input`, vybrané páry `selection` a vytvoří databázi nových molekul `output`. Parametr `-c` nastavuje maximální hledanou velikost hranového řezu, výchozí hodnota je 3.

5.5.1 Jednohranné křížení

Pro velikost 1 se v obou molekulárních grafech hledá taková společná hrana, jejíž odstranění by oba grafy rozdělilo na právě dvě komponenty. Spojením částí grafů opačných jedinců vznikne potomek.

5.5.1.1 Nalezení mostů grafu

Hrana, která graf dělí na dvě komponenty je též známa jako most grafu. Prvním krokem algoritmu je nalezení všech mostů v obou molekulárních grafech. Toho je dosaženo standardním algoritmem založeném na prohledávání do hloubky. Z nalezených mostů jsou vyřazeny takové, které propojují uzel s listem. Uchovávání takovýchto mostů nemá v kontextu křížení smysl, protože řezem a spojením této hrany by vznikl potomek identický s jedním ze dvou předchůdců.



Obrázek 5.2: Dvě molekuly nalevo podstupují jednohranné křížení. Zvolená společná hrana je vyznačena čárkovaně. Napravo jsou všechny čtyři možnosti nově složené molekuly.

5.5.1.2 Nalezení společných mostů

Společný most je takový, který má v obou grafech stejný typ (násobnost vazby) a propojuje uzly stejného typu (obsahují atomy se stejným symbolem). Následujícím krokem algoritmu je průchod a porovnání seznamů mostů obou grafů. Dvojice hran odpovídající výše uvedeným podmínkám jsou označeny za společné mosty.

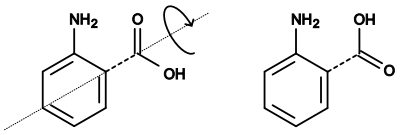
5.5.1.3 Provedení řezu

Každý společný most nabízí několik možností provedení řezu. Hrany, které propojují uzly rozdílného typu, umožňují křížením vytvořit dva rozdílné potomky. Uzly propojené takovou hranou označíme jako levý a pravý takovým způsobem, aby levé uzly v obou grafech byly stejného typu a pro pravé platilo totéž. Při zachování části prvního grafu s levým uzlem, části druhého grafu s pravým uzlem a následném propojení těchto částí, vznikne jeden z potomků. Při opačném označení uzlů (výměna levosti a pravosti) vznikne druhý potomek. Pokud hrana propojuje uzly stejného typu, existují další dvě možnosti jak označit uzly za levé či pravé. Tato označení odpovídají možným potomkům. Variantu s hranou propojující uzly stejného typu zachycuje obrázek 5.2.

5.5.1.4 Rotace podgrafu

Před samotným provedením řezu je nutné oba grafy posunout a otočit tak, aby se vybraná dvojice hran i jejich koncové uzly nacházely na stejné pozici v prostoru. Po těchto transformacích zůstává jeden stupeň volnosti – grafy je možné otáčet kolem osy hrany (osa prochází uzly, které hrana propojuje).

Pro získání rozumné geometrie potomka je nutné zvolit vhodnou rotaci kolem této osy. V ideálním případě je nalezena taková rotace, pro kterou existují ještě nějaké další dva uzly (mimo již dva ukotvené, propojené zvolenou hranou), které se prostorově kryjí. Pokud žádná taková rotace neexistuje, je nalezena nejbližší možná. Hledá se tedy taková orientace, kdy je vzdálenost nějakých dvou uzlů minimální. Aby po provedení řezu nedošlo k překryvu těchto uzlů, vybírají se vždy takové páry, aby jeden z uzlů byl z použité části jedné molekuly a druhý ze zahozené části druhé molekuly.



Obrázek 5.3: Ilustrace možnosti rotace grafu kolem osy zvolené hrany.

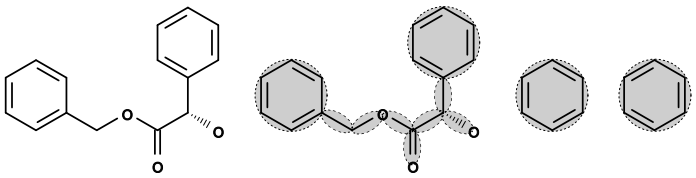
Ve většině případů budou vybrány uzly z nějaké struktury, která je pro obě molekuly společná. Zobrazením těchto bodů do sebe dojde k překryvu této struktury a po provedení řezu budou na sebe části plynule navazovat a potomek tak získá rozumnější geometrii, než kdybychom zvolili náhodnou rotaci.

5.5.2 Vícehranné křížení

Pokud je velikost hranového řezu větší než 1, je nutné zapojit i vícehranné křížení. K použití dojde, pokud je parametr `-n` nastaven na číslo 0 (značící libovolnou velikost hranového řezu), nebo číslo větší než 1. Pro účely tohoto algoritmu neuvažujeme takové řezy, které jsou nadbytečné. Nadbytečný je takový řez, ve kterém existuje hrana, po jejímž vyjmutí má daná množina hran stále vlastnost řezu.

5.5.2.1 Nalezení komponent grafu

Dalším zjednodušením je hledání řezů pouze v komponentách, které zbydou po odstranění veškerých mostů. Pro nalezení těchto částí je použit algoritmus pro hledání komponent 2-souvislosti. Tou je maximální podgraf pro který platí, že po odebrání libovolného uzlu zůstane souvislý. Mezi tyto komponenty patří i mosty, ty ale v případě vícehranného křížení nemá smysl uvažovat.



Obrázek 5.4: Nalezení 2-souvislých komponent grafu. Zleva doprava: jedna z vybraných molekul, nalezené a vyznačené komponenty, zbylé komponenty po vyřazení mostů.

5.5.2.2 Nalezení možných zobrazení

Pro všechny možné dvojice komponent z prvního a druhého grafu je nutné prozkoumat, zdali je možné jednu na druhou zobrazit. Podobně jako v případě

programu `LigMerge` hledáme takovou posloupnost posunů a rotací, aby došlo k překryvu co nejvíce uzlů a hran jednotlivých komponent. Pro dvě vybrané komponenty prozkoumáme všechny možnosti hrubou silou – každou hranu a uzel jedné komponenty se pokusíme zobrazit na každou hranu a uzel druhé komponenty. Možností je mnoho, jsou ale významně omezené následujícími podmínkami:

- Obě hrany musí být stejného typu.
- Všechny tři uzly (dva konce hrany plus jeden vybraný) z obou grafů musí mít odpovídající typy.
- Vzájemné prostorové vzdálenosti všech tří uzlů z obou grafů musí být stejné.

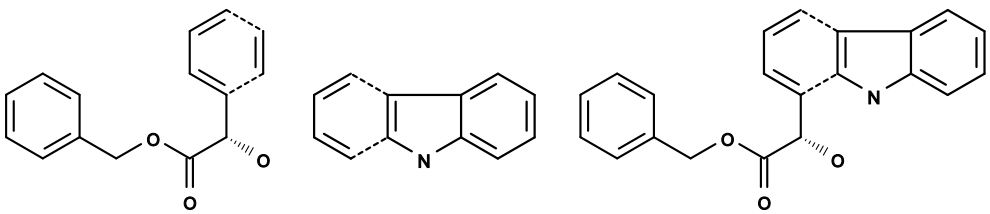
Z nalezených možností jsou nakonec odstraněny duplicity – přestože byly vybrány jiné hrany a uzly, mohlo by dojít ke stejnému zobrazení.

5.5.2.3 Průnik hran

Pro každé nalezené zobrazení je zjištěno, které z hran se prostorově kryjí. Takovýto průnik hran je největší společnou částí pro dané zobrazení. Z této množiny jsou následně vybírány podmnožiny, u kterých se zjišťuje vlastnost hranového řezu. Tato část je z celého programu výpočetně nejnáročnější, protože postupně dochází k výběru všech podmnožin velikostí 2 až c, kde c je maximální hloubka nastavená parametrem `-c`. Z nalezených řezů jsou vyřazeny takové řezy, které jsou nadbytečné (popsáno v podsekci 5.5.2).

5.5.2.4 Provedení řezu

Samotné použití řezu je jednodušší než u jednohranného křížení, neboť pozice a orientace obou grafů je pevně daná předem nalezeným zobrazením. Po výběru, která strana jakého grafu se má zachovat a která nahradit částí opačného grafu, jsou tyto části náležitě odpojeny od původních grafů a spojeny v nově vzniklého potomka.



Obrázek 5.5: Dvě molekuly nalevo podstupují vícehranné křížení. Zvolený hranový řez je vyznačen čárkovaně. Napravo je jeden z mnoha možných potomků.

5.6 Mutace

Po křížení je provedena mutace. Program se volá s následujícími argumenty:

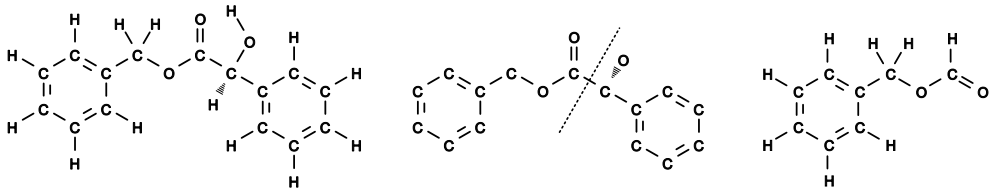
```
mutation [-m chance] input fragments output
```

V předchozím kroku získaní potomci jsou předáni v argumentu `input`; `fragments` je uživatelem dodaná databáze fragmentů, popřípadě molekuly z první generace. Zmutované molekuly jsou uloženy do výstupní databáze `output`.

Každá ze vstupních molekul je pro mutaci vybrána s pravděpodobností určenou parametrem `-m`. Pro vybrané molekuly je s 50% pravděpodobností provedeno zkrácení grafu, v opačném případě dojde k připojení fragmentu. Tento operátor pracuje pouze se zjednodušenými molekulárními grafy, neboli s grafy s odstraněnými uzly symbolizující atomy vodíku. Databáze zmutovaných molekul je následně doplněna o vodíky programem `OpenBabel`.

5.6.1 Zkrácení grafu

Jednodušší operací je zkrácení molekuly. Ze všech hran je náhodně vybrán hranový řez, který splňuje vlastnost ne-nadbytečnosti (podsekce křížení, 5.5.2). Po provedení řezu se největší vzniklá komponenta stává výslednou zmutovanou molekulou.



Obrázek 5.6: Mutace zkrácením grafu: Levý graf molekuly je zjednodušen na graf vprostřed. Po provedení řezu znázorněného čarou dojde k doplnění vodíků a vytvoření nového jedince vpravo.

5.6.2 Připojení fragmentu

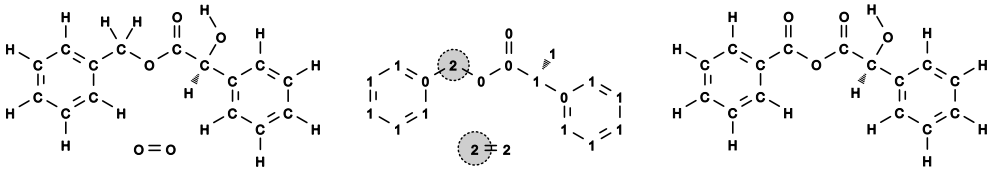
Náročnějším krokem mutace je připojení náhodně vybraného fragmentu k molekule. S pravděpodobností 50 % je fragment připojen uzlem, v opačném případě hranou.

5.6.2.1 Připojení uzlem

Pro každý uzel mutované molekuly je zjištěno, kolik k němu bylo připojeno vodíků, než došlo k zjednodušení. Každá hrana spojující uzel s uzlem vodíku

může být teoreticky nahrazena hranou stejného typu. Počet dříve připojených vodíků odpovídá maximálnímu počtu nově připojitelných vazeb. Násobné vazby (dvojná, trojná) pro připojení vyžadují ekvivalentní počet dříve připojených vodíků (2, 3).

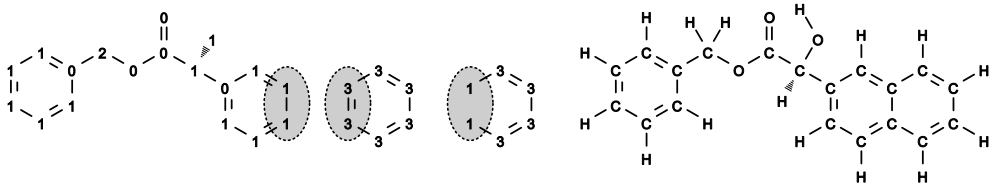
Je nalezena taková náhodná dvojice uzlů (jeden z mutované molekuly, jeden z připojovaného fragmentu), kde k uzlu mutované molekuly bylo dříve připojeno nejméně stejně vodíků, jako je součet typů hran připojených k uzlu fragmentu. Podobně jako u jednohranného křížení jsou grafy molekuly a fragmentu posunuty tak, aby se dvojice uzlů prostorově překrývala. Také jsou otočeny grafy pro zajištění překryvu bývalé hrany připojující jeden z vodíků a nově připojované hrany. Stále zbývá jeden stupeň volnosti, a to rotace kolem osy této hrany. Ta je zvolena náhodně, protože při doplňování vodíků programem `OpenBabel` je opravena geometrie celé molekuly. Nakonec dojde k sloučení této dvojice do jednoho uzlu, náležitému přepojení hran a výsledkem je nová zmutovaná molekula.



Obrázek 5.7: Mutace připojením fragmentu k uzlu: Levé grafy molekuly i fragmentu jsou zjednodušeny na grafy vprostřed. Čísla v uzlech mutované molekuly značí počet dříve připojených vodíků. Čísla v uzlech fragmentu označují součet typů připojených hran. Je nalezena dvojice uzlů (vyznačená kruhy) pro kterou platí, že číslo v uzlu mutovaného grafu je větší nebo stejné jako číslo v uzlu fragmentu. Dojde k sloučení uzlu, přepojení hran, doplnění vodíků a výsledkem je nový jedinec vpravo.

5.6.2.2 Připojení hranou

Podobně probíhá připojení fragmentu hranou. Namísto dvojice uzlů je vybrána náhodná dvojice hran, která splňuje podobné podmínky vztahující se na koncové uzly vybraných hran. Porovnává se počet dříve připojených vodíků se součtem typů nově připojovaných hran, kromě hrany vybrané do dvojice. Grafy jsou posunuty a otočeny takovým způsobem, aby došlo k překryvu vybrané dvojice hran a dvou jejich koncových uzlů. Zbývající stupeň volnosti rotace kolem osy hrany je ukotven výběrem jednoho z dříve připojených vodíků, nově připojované hrany a jejich prostorovým zarovnáním. Posledním krokem je sloučení vybrané dvojice hran, příslušných koncových uzlů a následné přepojení hran.



Obrázek 5.8: Mutace připojením fragmentu k hraně: Levý graf mutované molekuly i fragmentu jsou již zjednodušeny a očíslovány. Je vybrána dvojice hran (vyznačená elipsami), která splňuje podmínku, že čísla v obou uzlech mutovaného grafu jsou stejná nebo větší než čísla v uzlech fragmentu po odečtení typu vybrané hrany fragmentu. Následující graf fragmentu (druhý zprava) ilustruje odečtení vybrané hrany a splnění podmínky pro výběr dvojice. Po sloučení hrany a doplnění vodíků vzniká pravý graf nového jedince.

5.7 Začlenění

Posledním operátorem genetického algoritmu je začlenění. Zkřížené molekuly společně se zmutovanými a opravenými molekulami prošly ohodnocením a nyní je nutné vybrat, kteří jedinci budou tvořit novou populaci. Volání je následující:

```
inclusion [-e ratio] [-t size] elders elders_eval offsprings
    ↪ offsprings_eval output output_eval
```

Úkolem tohoto operátoru je sloučit předchozí populaci `elders` s nově vzniklými potomky `offsprings`, a vytvořit populaci nové generace `output`. Kromě databází molekul dojde také k sloučení ohodnocení populací `elders_eval` a `offsprings_eval` do souboru pro novou generaci `output_eval`.

Výběr jedinců nové generace probíhá následujícím způsobem. Nejprve je z obou populací vybrána elita, nejlepší jedinci, o které nechceme přijít. Ze všech jedinců, seřazených podle jejich ohodnocení je oddělena vrchní část a použita jako základ pro novou populaci. Velikost této vrchní části lze ovlivnit parametrem `-e`, který určuje poměr mezi elitou a zbytkem populace. Výchozí poměr je 0.5. Tento poměr se vztahuje k výstupní generaci, pro počáteční populaci o 100 jedincích bude elita tvořena 50 jedinci.

Kapitola 6

Testování

Program byl úspěšně nainstalován a spuštěn společně se softwarem `vms` a dalšími závislostmi[6] na následujících systémech:

- Gentoo 4.4.6 (x86_64)
- MacOS 10.6.8 (x84_64)
- Cygwin 2.7.0 (x86_64)

Program byl úspěšně nainstalován a spuštěn bez softwaru `vms`[7] na následujících systémech:

- OpenBSD 6.1 (amd64)
- OpenBSD 6.1 (armv7)
- NetBSD 7.0.2 (i386)
- FreeBSD 11.0 (i386)

Veškeré výpočetní testy byly prováděny v laboratoři SAGElab [22]. Vstupní databází pro genetický algoritmus byl seznam molekul použitý v článku *Computational Design and Selection of Optimal Organic Photovoltaic Materials* [10]. Požadavek pro `vms` je vidět na obrázku 6.1. Jedná se o upravené zadaní z práce *Distribuovaná infrastruktura pro virtuální screening molekul* [11]. Hodnotí molekuly na základě intenzity absorbce či emise elektromagnetického záření v zadaném intervalu vlnových délek (0–800 nm).

Byly provedeny tři testy pro velikosti turnaje 1, 2 a 3. Jednotlivé běhy se tak odlišovaly mírou selekčního tlaku – čím větší turnaj, tím lépe hodnocení jedinci budou vybíráni při selekci a začlenění; menší turnaj naopak dává příležitost i méně vhodným jedincům.

[6] `ORCA` (pro software `vms`) a `OpenBabel`.

[7] Pro tyto architektury a systémy neexistují binární spustitelné soubory softwaru `ORCA`, na kterém `vms` závisí. Program byl testován se zástupným ohodnocovacím backendem `fake_vms`.

```
# General mandatory directives
job excited
method BP86

# General optional directives
basis def2-SV(P)
memory 2048

# Job specific mandatory directives
nroots 10

# Job specific optional directives
iroot 3
maxdim  150

# Job specific optional filters
absorb 0 800 0.001 1.00 1.00
```

Obrázek 6.1: Požadavek pro `vms` použitý při testování.

Počáteční populace obsahovala 132 molekul a pro každý test bylo vypočteno 100 generací[8]. Po odfiltrování duplicitních a dříve spočtených molekul bylo pomocí `vms` celkem ohodnoceno přes 16 tisíc molekul. Výpočet trval více než 150 hodin, doba výpočtu jedné generace byla v průměru 30 minut[9]. Ohodnocení jedné molekuly v průměru trvalo kolem 33 sekund[10]. Režie `mm` při výpočtu jedné generace se v průměru pohybovala kolem 0.3 %[11].

Podíl na době výpočtu ohodnocení jedné generace je možné vidět z grafu 6.2. Jedná se o data ze všech tří testů. Z průměru těchto dat jsou vytvořeny grafy 6.3 a 6.4. Data z prvních několika generací jsou silně zkreslena použitím fitness cache, protože při prvním běhu bylo nutné ohodnotit celou počáteční generaci. Při následujících testech bylo toto ohodnocení již uloženo, tudíž nebylo nutné tyto molekuly znovu ohodnocovat. Kolem desáté generace došlo k zániku tohoto efektu, protože jednotlivé populace byly již dostatečně odlišné.

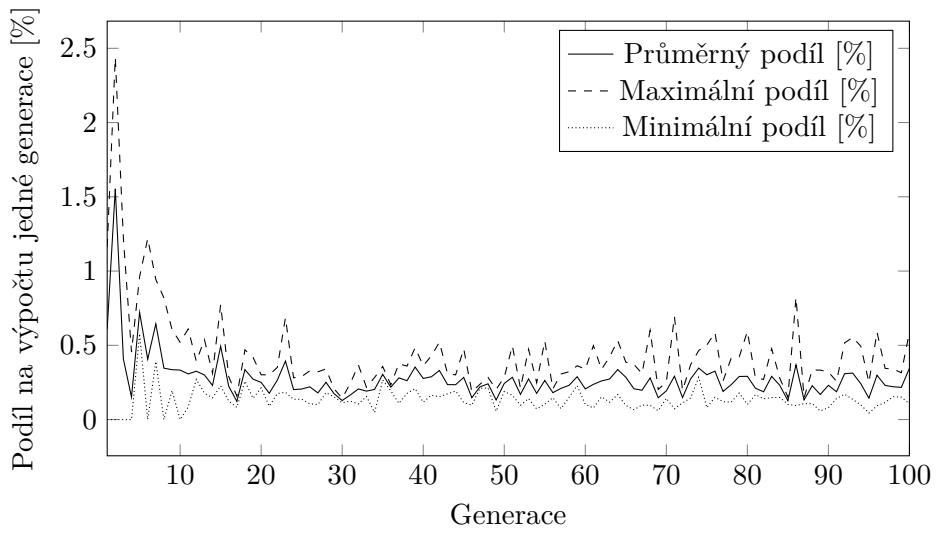
Na grafech 6.6 až 6.9 je zobrazen vývoj fitness vybraných jedinců v průběhu generací. Sledováno bylo hodnocení nejlepšího jedince, středního jedince (medián) a průměrného jedince (aritmetický průměr).

[8] Tento počet byl zvolen po spočtení 168 generací prvního testu, neboť od 100. generace nedocházelo k výrazným změnám.

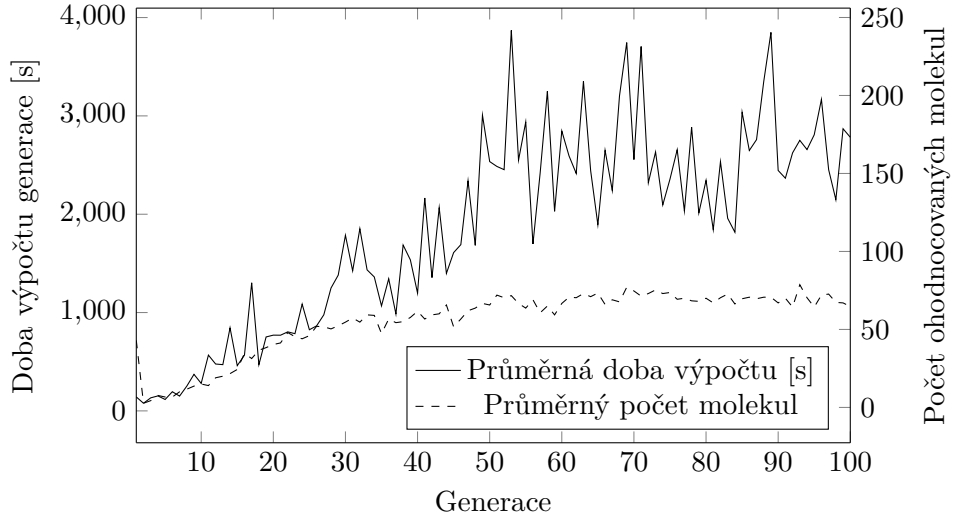
[9] Nejkratší výpočet trval 40 sekund, nejdelší 115 minut.

[10] Při rozpočítání průměrné doby výpočtu mezi průměrný počet ohodnocovaných molekul v jedné generaci. Jednotlivá ohodnocení trvala od 4 sekund do 112 minut. Průměrná doba ohodnocení jedné molekuly byla 556 sekund. Výpočet ale probíhal paralelně (v 32 běžích), proto je rozpočítaná průměrná doba výrazně nižší.

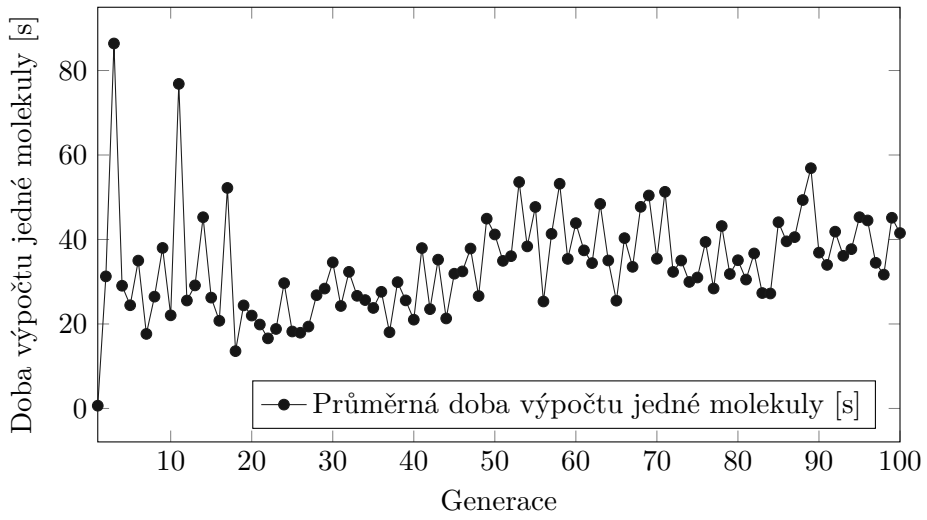
[11] Nejvyšší režie činila 2.5 %, nejnižší 0 % (měření probíhalo v sekundách a výpočet `mm` proběhl v čase kratším než jedna sekunda).



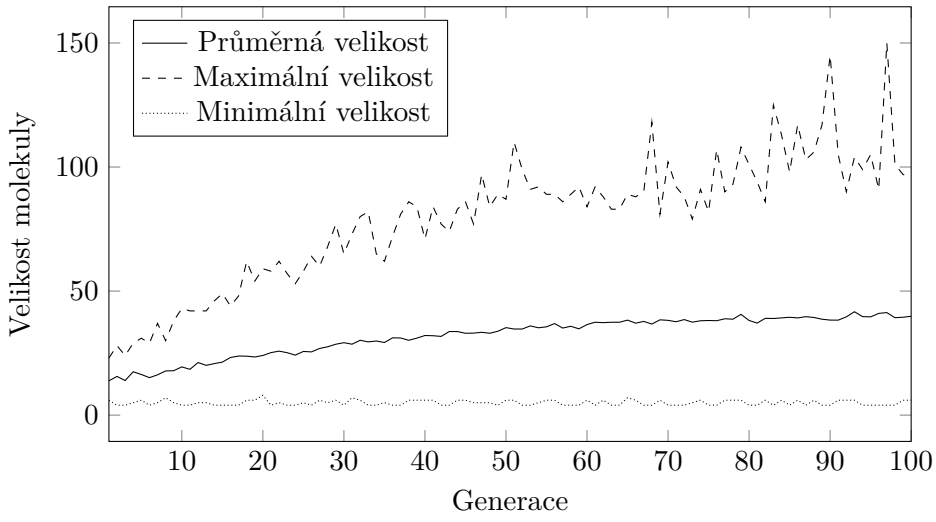
Obrázek 6.2: Podíl `mm` na době běhu výpočtu jedné generace. Průměrný podíl se pohybuje kolem 0.3 %.



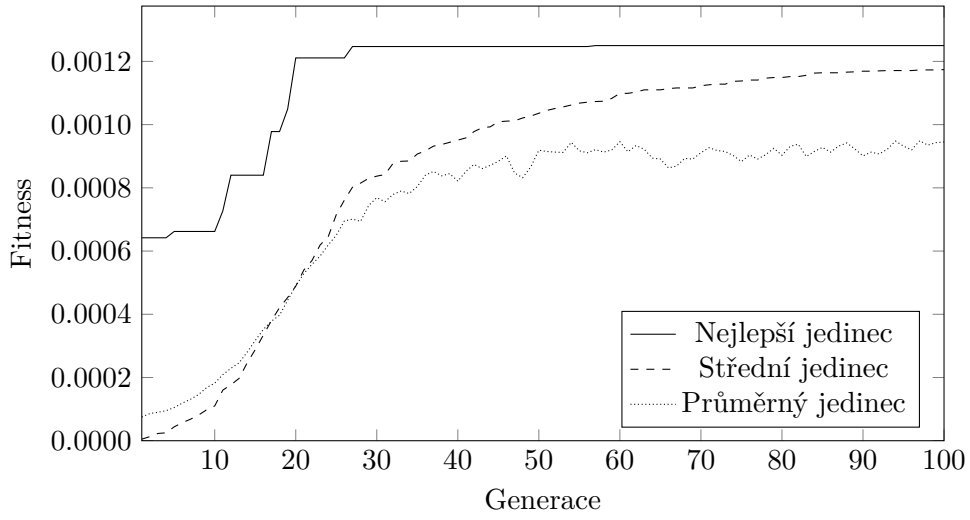
Obrázek 6.3: V první čtvrtině grafu je vidět jistá závislost průměrné doby výpočtu jedné generace na průměrném počtu ohodnocovaných molekul. Během zbylých generací se doba nutná k ohodnocení jedné molekuly zvyšuje a výrazně kolísá, jak dochází k ohodnocování molekul rozdílné složitosti (graf 6.5).



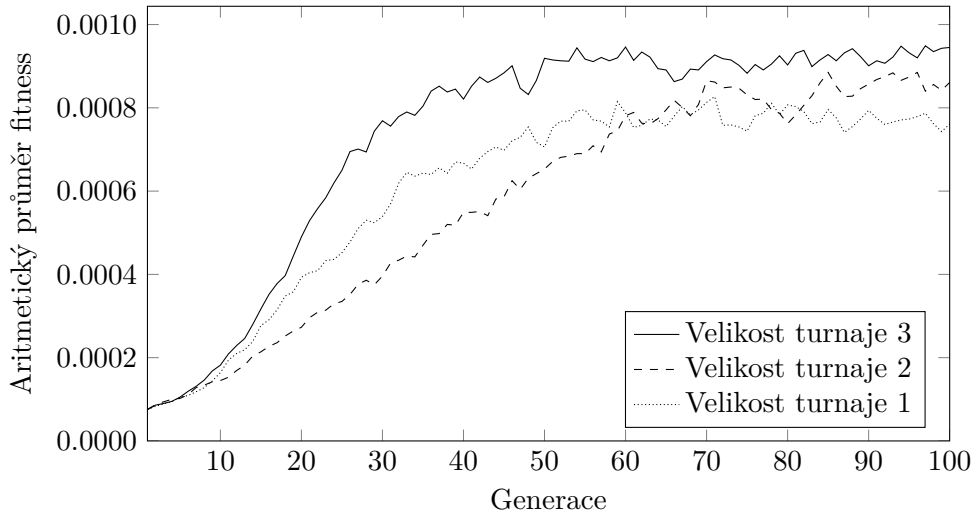
Obrázek 6.4: Průměrná doba výpočtu jedné molekuly. Podobně jako v grafu 6.3 je vidět mírný trend zvyšování délky výpočtu v závislosti na generaci.



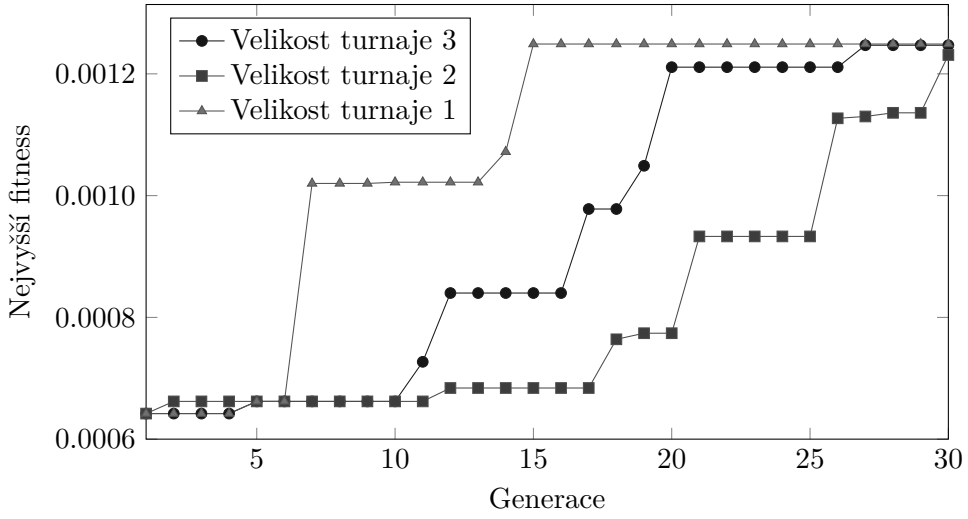
Obrázek 6.5: Doba ohodnocení molekuly závisí na mnoha faktorech. Nejsnáze měřitelným faktorem je velikost molekuly v atomech. Z prvotní populace, kde největší molekula obsahuje 23 atomů, se postupně vytváří složitější molekuly. Největší ohodnocená molekula obsahovala 145 atomů.



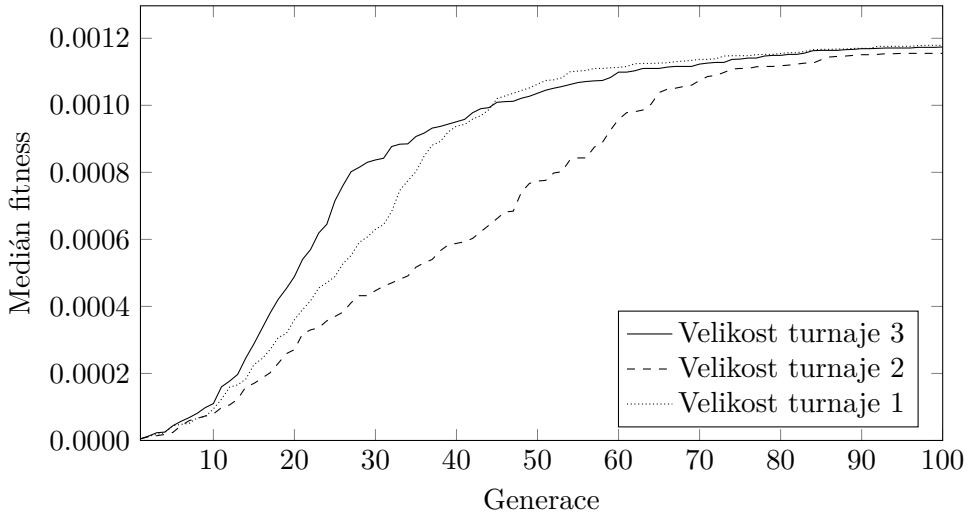
Obrázek 6.6: Graf ohodnocení nejlepšího, středního a průměrného jedince pro velikost turnaje 3.



Obrázek 6.7: Graf ohodnocení průměrných jedinců pro velikost turnaje 1, 2 a 3. Při běhu s velikostí turnaje 1 došlo k brzkému šťastnému nálezu dobře hodnoceného jedince (graf 6.8). Díky tomu až do poslední čtvrtiny neodpovídá průměrné hodnocení velikosti turnaje. V pravé části grafu se průměrné hodnocení ustálilo v rozmezích odpovídajících velikosti turnaje.



Obrázek 6.8: Graf ohodnocení nejlepších jedinců v průběhu prvních 30 generací pro velikost turnaje 1, 2 a 3. Jak již bylo zmíněno u grafu 6.7, při běhu s velikostí turnaje 1 byl nečekaně rychle nalezen dobře hodnocený jedinec. Po zbylých 70 generací nedochází k výrazné změně nejlepšího jedince.



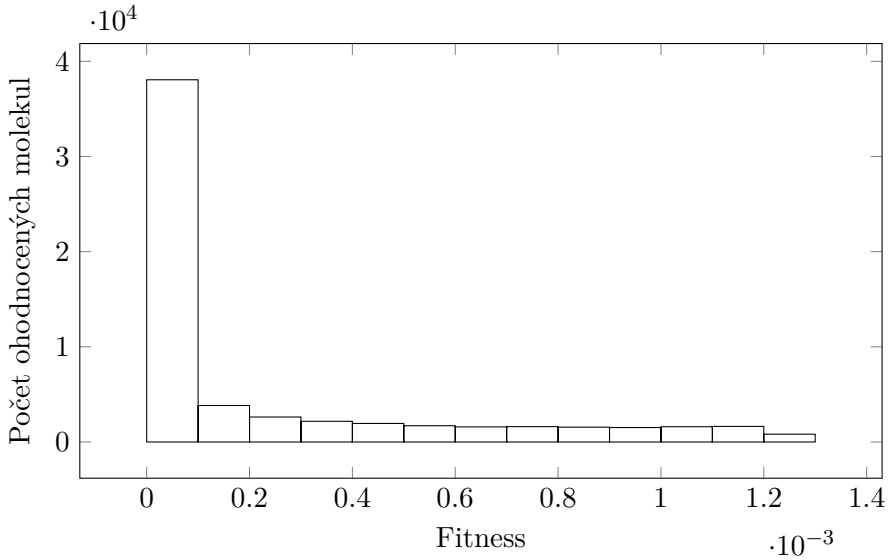
Obrázek 6.9: Graf ohodnocení středních jedinců pro velikost turnaje 1, 2 a 3. Lepší výsledek velikosti turnaje 1 je opět způsoben dříve zmíněným brzkým nálezem.

Při výše zmíněných testech i při testování během vývoje byl využíván jeden soubor jako společná fitness cache. Po skončení testování tento soubor obsahoval přes 60 tisíc záznamů o ohodnocených molekulách. Před implementací cache bylo ohodnoceno kolem 43 tisíc molekul. V laboratoři SAGElab tak bylo ohodnoceno více než 100 tisíc molekul. Z dat uložených ve fitness cache byl vytvořen graf 6.10. Uložené hashe molekul s fitness vyšší než 1.2×10^{-3} (celkem 822) byly vyhledány v databázi PubChem [23]. Byly nalezeny záznamy pro 5 z těchto molekul (obr. 6.11) a 3 látky byly i komerčně dostupné.

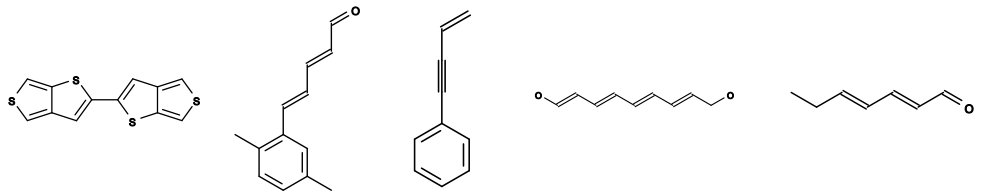
Program se osvědčil pro svůj účel – generování rozsáhlé sady molekul, odpovídající uživatelskému zadání. Oproti jiným oblastem využití genetického algoritmu, kde nás často zajímá pouze nejlepší jedinec, zužitkujeme celou elitu po několik generací. Část z vygenerovaných molekul se může ukázat jako nevhodná. Ať už stabilitou, dostupností, využitelností pro daný problém, nebo také nemusí vůbec být syntetizovatelná. Virtuální screening rozhodně označil více jak polovinu kandidátů (graf 6.10) jako jedince naprosto nevyhovující zadání uživatele. Došlo tak k jasnému vymezení, které vygenerované molekuly má vůbec cenu uvažovat jako možná řešení uživatelova problému.

V grafu 6.6 vidíme postupně přibližujícího se středního jedince k nejlepšímu jedinci. Ve 24. generaci překonal nejlepšího jedince z první generace a společně s ním ho také překonala celá elita, v našem případě vrchní polovina populace. Pokud by uživateli stačily molekuly s fitness vyšší než 1×10^{-3}, od 45. generace do této kategorie spadá celá elita. Každou následující generaci má uživatel na výběr z více než 66 molekul[12].

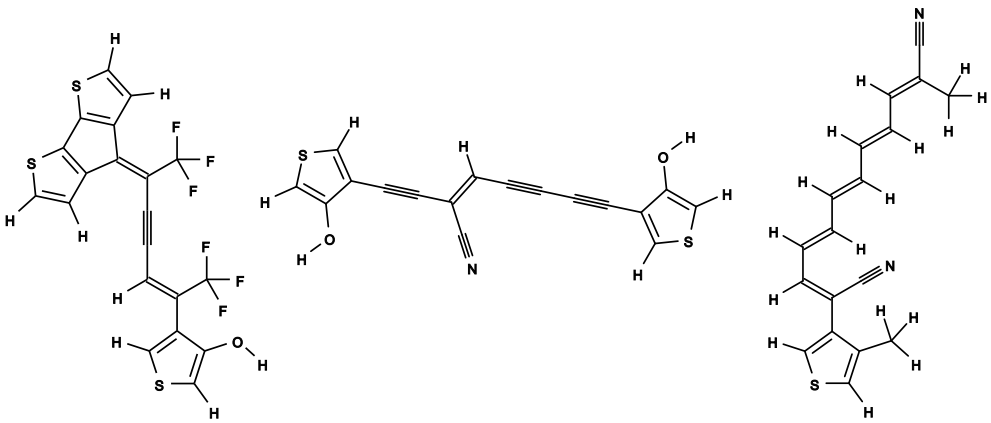
[12]Elita o 66 jedincích a další jedinci splňující podmínku uživatele.



Obrázek 6.10: Histogram počtu ohodnocených molekul pro danou fitness. Více než polovina jedinců měla fitness blízkou nule.



Obrázek 6.11: Molekuly s fitness vyšší než 1.2×10^{-3} s existujícím záznamem v databázi PubChem [23].



Obrázek 6.12: Tři nejlepší nalezené molekuly, s fitness vyšší než 1.25×10^{-3}.

Závěr

Cílem práce bylo vytvořit program pro generování molekul na základě požadavků uživatele. Na základě rešerše byla vybrána vhodná reprezentace molekuly pro genetický algoritmus a příslušně navrženy genetické operátory křížení a mutace. Pro ohodnocení molekul byl využit software pro virtuální screening kolegy Štěpána Sršně.

Výsledkem je program, který uživateli umožňuje vygenerovat rozsáhlou sadu molekul odpovídající zadaným požadavkům pro virtuální screening. Všechny operátory genetického algoritmu byly implementovány v jazyce C. Nad těmito programy byl vybudován shellový skript zajišťující běh celého algoritmu. Program je tak jednoduše rozšiřitelný přidáním nebo úpravou nějakého z operátorů, nebo změnou běhu celého genetického algoritmu. Program byl otestován na různorodých unixových systémech, aby byla zajištěna jeho přenositelnost. Veškeré spustitelné soubory byly náležitě zdokumentovány formou standardních manuálových stránek.

Genetický algoritmus by bylo možné do budoucna doplnit o další metody selekce a začlenění. Také by šlo implementovat vybranou formu nichingu – znevýhodnění geometricky nebo strukturně podobných jedinců za účelem úniku z lokálního maxima.

Dalším možným vylepšením by byla paralelizace běhu celého genetického algoritmu. V aktuální implementaci probíhá paralelně pouze ohodnocení molekul a pro postup do další generace je nutné vyčkat na všechna hodnocení. Rozdělením genetického algoritmu na několik paralelních běhů by došlo k plnému využití výpočetních prostředků. Jednalo by se o implementaci ostrovního modelu, kdy algoritmus probíhá na několika oddělených ostrovech (menší samostatné skupiny jedinců) a po uplynutí určitého počtu generací dojde k promíchání jedinců mezi jednotlivými ostrovy. K synchronizaci běhů by tak docházelo pouze před touto výměnou jedinců.

Pro pohodlí uživatele by také bylo možné přidat grafické prostředí, které by umožňovalo měnit parametry genetického algoritmu nebo vizualizovalo jeho průběh.

Literatura

[1] Holland, J. H.: *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press, 1975.

[2] Norvig, P.; Russell, S. J.: *Artificial Intelligence: A Modern Approach (Third Edition).* Prentice Hall, Inc., 2009, ISBN 978-0136042594.

[3] NASA: ST5. [online], 2007, [cit. 20. 4. 2017]. Dostupné z: `https://www.nasa.gov/mission_pages/st-5/main/index.html`

[4] Lohn, J. D.; Hornby, G. S.; Linden, D. S.: An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission. Chapter 1. Technická zpráva, NASA Technical Reports Server, 2004. Dostupné z: `https://ti.arc.nasa.gov/m/pub-archive/812h/0812%20(Lohn).pdf`

[5] Durrant, J. D.; Lindert, S.; McCammon, J. A.: AutoGrow 3.0: An improved algorithm for chemically tractable, semi-automated protein inhibitor design. *Journal of Molecular Graphics and Modelling*, ročník 44, 2013: s. 104–112, doi:10.1016/j.jmgm.2013.05.006. Dostupné z: `http://autogrow.ucsd.edu/`

[6] Chéron, N.; Jasty, N.; Shakhnovich, E. I.: OpenGrowth: An Automated and Rational Algorithm for Finding New Protein Ligands. *Journal of Medicinal Chemistry*, ročník 59, č. 9, 2016: s. 4171–4188, doi:10.1021/acs.jmedchem.5b00886. Dostupné z: `https://opengrowth.sourceforge.io/`

[7] Westhead, D. R.; Clark, D. E.; Frenkel, D.; aj.: PRO_LIGAND: An approach to de novo molecular design. 3. A genetic algorithm for structure refinement. *Journal of Computer-Aided Molecular Design*, ročník 9, č. 2, 1995: s. 139–148, doi:10.1007/bf00124404.

[8] Gillet, V.; Johnson, A. P.; Mata, P.; aj.: SPROUT: A program for structure generation. *Journal of Computer-Aided Molecular Design*, ročník 7, č. 2, 1993: s. 127–153, doi:10.1007/bf00126441.

[9] Shu, Y.; Levine, B. G.: Simulated evolution of fluorophores for light emitting diodes. *The Journal of Chemical Physics*, ročník 142, č. 10, 2015: str. 104104, doi:10.1063/1.4914294.

[10] O'Boyle, N. M.; Campbell, C. M.; Hutchison, G. R.: Computational Design and Selection of Optimal Organic Photovoltaic Materials. *The Journal of Physical Chemistry C*, ročník 115, č. 32, 2011: s. 16200–16210, doi:10.1021/jp202765c.

[11] Štěpán Sršeň: *Distribuovaná infrastruktura pro virtuální screening molekul*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[12] Koza, J.: *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, Mass: MIT Press, 1992, ISBN 978-0262111706.

[13] Luke, S.: *Essentials of Metaheuristics*. Lulu, druhé vydání, 2013, ISBN 978-1300549628. Dostupné z: `http://cs.gmu.edu/~sean/book/metaheuristics/`

[14] Wong, S. S. Y.; Luo, W.; Chan, K. C. C.: EvoMD: An Algorithm for Evolutionary Molecular Design. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, ročník 8, č. 4, 2011: s. 987–1003, doi:10.1109/tcbb.2010.100.

[15] Trinajstić, N.: *Chemical graph theory*. Boca Raton: CRC Press, 1992, ISBN 978-0849342561.

[16] Open Babel: XYZ (format). [online], 2007, [cit. 5. 5. 2017]. Dostupné z: `http://openbabel.org/wiki/XYZ_(format)`

[17] Accelrys Software Inc.: *CTfile Formats*. 2011. Dostupné z: `http://download.accelrys.com/freeware/ctfile-formats/ctfile-formats.zip`

[18] National Center for Biotechnology Information: Water. [online], 2017, [cit. 30. 4. 2017]. Dostupné z: `https://pubchem.ncbi.nlm.nih.gov/compound/962`

[19] Lindert, S.; Durrant, J. D.; McCammon, J. A.: LigMerge: A Fast Algorithm to Generate Models of Novel Potential Ligands from Sets of Known Binders. *Chemical Biology & Drug Design*, ročník 80, č. 3, 2012: s. 358–365, doi:10.1111/j.1747-0285.2012.01414.x. Dostupné z: `http://rocce-vm0.ucsd.edu/data/sw/hosted/ligmerge/`

[20] O'Boyle, N. M.; Banck, M.; James, C. A.; aj.: Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, ročník 3, č. 1, 2011: str. 33, doi:10.1186/1758-2946-3-33. Dostupné z: `http://openbabel.org/`

[21] Stein, S. E.; Heller, S. R.; Tchekhovskoi, D. V.; aj.: *IUPAC International Chemical Identifier (InChI)*. IUPAC, 2017. Dostupné z: `http://www.inchi-trust.org/download/105/INCHI-1-DOC.zip`

[22] CESNET, z.s.p.o.: SAGElab. [online], 2017, [cit. 5. 5. 2017]. Dostupné z: `https://sagelab.cesnet.cz/`

[23] National Center for Biotechnology Information: PubChem Compound Database. [online], 2017, [cit. 5. 5. 2017]. Dostupné z: `https://pubchem.ncbi.nlm.nih.gov/`

Příloha **A**

Obsah přiloženého CD

- `readme.txt` ..... popis obsahu CD a postup instalace programu
- `thesis.pdf` ..... text práce ve formátu PDF
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX