

Univerzita Karlova v Praze
Přírodovědecká fakulta

BAKALÁŘSKÁ PRÁCE



Miloš Halda

Počítačové simulace struktury proteinů pomocí zhrubených modelů

Katedra fyzikální a makromolekulární chemie

Vedoucí bakalářské práce: Ing. Lucie Nová, Ph.D.

Studijní program: Bioinformatika

Studijní obor: B-BINF

Praha 2022

Děkuji všem blízkým lidem, kteří mě podporovali během mého studia i během psaní této práce. Chci poděkovat také za vedení a odborné zázemí svojí vedoucí práci, Lucii Nové, která mi ochotně dávala nové podněty a zpětnou vazbu k mojí práci. Také chci poděkovat výzkumnému týmu, především Markétě Pavlíkové, se kterou jsme (minimálně pocitově) dlouhé hodiny řešili způsob výpočtu dihedrálního úhlu a potenciálu.

Výpočetní zdroje byly dodány projektem ”e-Infrastruktura CZ”(e-INFRA CZ LM2018140) podpořeného Ministerstvem školství, mládeže a tělovýchovy České republiky.¹

¹Přeloženo z anglického originálu: ”Computational resources were supplied by the project ”e-Infrastruktura CZ”(e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.”

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

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

Tato práce ani její podstatná část nebyla předložena k získání jiného nebo stejného akademického titulu.

V dne

Podpis autora

Název práce: Počítačové simulace struktury proteinů pomocí zhrubených modelů

Autor: Miloš Halda

Katedra: Katedra fyzikální a makromolekulární chemie

Vedoucí bakalářské práce: Ing. Jméno Vedoucí, Ph.D., Katedra fyzikální a makromolekulární chemie

Abstrakt:

Hlavní částí této bakalářské práce je funkční program pro simulace zhrubených modelů proteinů, vhodný pro testování funkcí potenciálu. Program využívá návrh stavu pomocí pivota a vyhodnocení nových stavů metodou Monte Carlo. Program také umožňuje využít metodu simulovaného žíhání s lineárním poklesem teploty a má zabudované základní nástroje pro vyhodnocení simulací (počítání průměrů a chyb blokovou metodou). Program byl otestován simulací několika reálných proteinů. Pro kratší proteiny (88 a méně aminokyselin) program fungoval podle očekávání, delší proteiny (111 a více aminokyselin) poukázaly na limity přístupu návrhu stavů pomocí pivota. Pro další vývoj bylo navrženo rozšíření možností návrhu nového stavu pro vylepšení simulací dlouhých řetězců.

Klíčová slova: simulace, struktura proteinu, zhrubený model, Monte Carlo

Title: Computer simulations of protein structures using coarse-grained models

Author: Miloš Halda

Department: Department of Physical and Macromolecular Chemistry

Supervisor: Ing. Lucie Nová, Ph.D., Department of Physical and Macromolecular Chemistry

Abstract:

The main part of this bachelor's thesis is an operational software for coarse-grained protein simulations, suitable for testing of new potential functions. The program is using a pivot-based proposal of new states and Monte Carlo evaluation of the proposed state. The program also allows to use a simulated annealing technique with the linear temperature decrease. Mean estimation and error estimation using the Block Method are implemented for evaluation of the simulations. The software was tested on several proteins. The simulations provided expected results for shorter chains (88 and less aminoacids), but simulations of longer chains (111 and more aminoacids) have shown the software limitations. Several options for the software improvement regarding the new state proposal for simulations of longer chains were discussed.

Keywords: simulation, protein structure, coarse-grained model, Monte Carlo

Obsah

1 Úvod	3
1.1 Homology based a <i>de novo</i> přístupy	3
1.2 Některé metody v rámci <i>de novo</i> přístupu	4
1.2.1 Molekulová dynamika	4
1.2.2 Monte Carlo	4
1.2.3 Hamiltoniánské Monte Carlo	5
1.3 Přístup Alpha Foldu a jeho význam	5
1.4 Cíle práce	5
2 Metodika	6
2.1 Stručný popis algoritmu simulace	6
2.1.1 Inicializace	6
2.1.2 Běh simulace	6
2.1.3 Finalizace	6
2.2 Zhrubařné modely	7
2.2.1 Model použitý v této práci	7
2.3 Potenciály	8
2.3.1 Nevazebný potenciál	9
2.3.2 Vazebný potenciál	11
2.3.3 Bending potenciál	11
2.3.4 Dihedrální potenciál	12
2.3.5 Boltzmannova inverze	13
2.4 Návrh stavu	14
2.4.1 Návrh stavu pomocí pivota	14
2.5 Monte Carlo vyhodnocení stavu	15
2.5.1 Boltzmannův faktor	16
2.5.2 Simulované žíhání	16
2.6 Simulační program	16
2.6.1 Programovací jazyk	17
2.6.2 Datové struktury, třídy a metody	17
2.6.3 Vstupy a výstupy	19
2.7 Jednotky simulace	20
2.7.1 Délka	21
2.7.2 Hmotnost	21
2.7.3 Teplota	21
2.7.4 Boltzmannova konstanta	21
2.7.5 Energie	21
2.7.6 Čas	21

2.8	Vyhodnocení simulace	23
2.8.1	Průměry	23
2.8.2	Chyby	23
2.8.3	RMSD	23
3	Výsledky a diskuze	24
3.1	Simulační program	24
3.2	Testování simulačního programu	24
3.2.1	Výběr proteinů pro simulace	24
3.2.2	Parametry zadанé simulace	25
3.2.3	Výsledky simulací	26
4	Závěry	30
	Literatura	33
	Seznam obrázků	36
	Seznam tabulek	37
	Přílohy	38
A	Fasta sekvence zkoumaných proteinů	39

Kapitola 1

Úvod

Proteiny jsou pro život jedny z nejpodstatnějších biopolymerů s velmi širokým spektrem funkcí. Základní součástí každého proteinu je polypeptidové vlákno složené z 20 (22) typů aminokyselin. Struktura proteinů má několik úrovní. Z primární struktury (tedy sekvence aminokyselin) ještě není možné snadno určit funkci daného proteinu i přesto, že máme dobrou znalost fyzikálně chemických vlastností jednotlivých aminokyselin.

Funkce proteinu je určena především sekundární, terciální a kvartérní strukturou proteinu. V současné době ještě není známé úplné řešení pro to, jak odhadnout sekundární a terciární strukturu proteinů pouze na základě jejich primární struktury.¹ Prostorová struktura proteinů je proto standardně zkoumána pomocí metod, založených na pozorování a experimentu (nejčastěji krystalografie) a zároveň s tím jsou vyvíjeny stále přesnější metody simulace, které se snaží zjistit strukturu proteinů bez nutnosti experimentu provádět.

Počítačové simulace se nepoužívají pouze na zjišťování prostorové struktury proteinů. Například práce Beyer 2022 [2] využívá simulace pro studium pohybů v hydrogelech v závislosti na pH roztoku a koncentraci rozpuštěných solí.

Co se týká proteinů, simulační programy pro odhad prostorové struktury proteinů se opírají o tzv. Anfinsenovo dogma [3], tedy přesvědčení, že prostorovou strukturu proteinů je možné určit pouze ze znalosti jejich primární struktury. Tato myšlenka je v současnosti někdy kritizována. Například sulfidické vazby v rámci proteinu vznikají ještě před dokončením sekundární a terciární struktury proteinů.² To znamená, že tyto vazby jsou důležité už při formování sekundární a terciární struktury. Nemají tak pouze stabilizační funkci, jak dříve předpokládalo.

Pro lepsí vysvětlení obsahu této práce je vhodné ukázat základní principy v často využívaných metod.

1.1 Homology based a *de novo* přístupy

Pro zjišťování sekundární a terciární struktury proteinů z jeho sekvence se používají dva základní přístupy.

Prvním je tzv. "homology based přístup". V rámci něj se prostorové uspořádání neznámého proteinu odhaduje na základě podobnosti s primárními strukturami

¹Toto téma je diskutováno například v review Dill 2008 [1].

²Gambardella 2022 [4].

známých proteinů.³

Druhým základním přístupem je tzv. *de novo*. Ten využívá znalost primární struktury proteinu (sekvence aminokyselin) a na základě toho se snaží získat jeho prostorovou strukturu. Hlavní myšlenkou *de novo* přístupu je efektivně prohledat prostor všech konformací proteinu a získat z něj tu nejpravděpodobnější. Vzhledem k obrovskému množství možných konformací je *de novo* přístup zpravidla implementován tak, že do určité míry napodobuje proces skládání proteinů v reálných systémech. Proto jsou vyvíjeny fyzikální modely, které napodobují chování skutečných proteinů, včetně energetických polí, sil a vazeb.

Dále jsou popsány konkrétní metody simulace v rámci *de novo* přístupu.

1.2 Některé metody v rámci *de novo* přístupu

1.2.1 Molekulová dynamika

Jedna z nejznámějších metod na prozkoumávání konformačního prostoru je molekulová dynamika. Simulace pomocí molekulové dynamiky se snaží co nejpřesněji napodobit pohyb částic reálných systémů. V každém kroku jsou aktualizovány pozice jednotlivých částic na základě silových polí v systému.⁴ Příkladem programu využívajícího molekulovou dynamiku je ESPResSo [7]. ESPResSo umožňuje mimo jiné definovat vlastní systém jednotek, pracovat s různými typy zhrubených modelů. Je volně přístupné a spravované Institutem pro výpočetní fyziku Univerzity ve Stuttgartu pod licencí GNU General Public Licence (GPL).

1.2.2 Monte Carlo

Další širokou skupinou metod jsou Monte Carlo simulace, tedy modely pracující s určitou mírou náhody. Do této kategorie patří i tato práce. Od sebe se jednotlivé Monte Carlo metody liší ve způsobu reprezentace proteinu a ve způsobu návrhu nových stavů simulace. V této práci byla pro návrh nového stavu použita metoda náhodného pootočení řetězce okolo pivotu.

Výhodou principu Monte Carlo je, že díky přijímání nových stavů s určitou mírou náhody nemusí metoda návrhu nového stavu nutně napodobovat pohyb molekul v reálném systému. Nevhodné, a tedy nepravděpodobné stavy (např. pokud se dvě částice překrývají) jsou totiž v dobře nastavené simulaci odstraněny během Monte Carlo vyhodnocení.

Využívat Monte Carlo k molekulovým simulacím navrhl N. Metropolis v roce 1953 [8]. Jeho princip dále popisuje Frenkel a Smit 2002 [9]. Současnými příklady využití přístupu Monte Carlo jsou například práce Wilson 2022 [10] a Neamtu 2023 [11]. Konkrétně v případě Neamtu 2023 byla studována vazebná afinita monoklonálních protilátek k vazebné doméně spike proteinů SARS-CoV-2 s využitím zhrubených modelů proteinu a Monte Carlo přístupu.

³Například práce Zhang 2005 [5] v úvodu uvádí základní principy "homology based" modelování prostorové struktury proteinů.

⁴Podrobnosti o molekulové dynamice např. zde: [6].

1.2.3 Hamiltoniánské Monte Carlo

Přístup Monte Carlo a molekulová dynamika se protínají v Hamiltoniánském Monte Carlu. V rámci této metody je na základě molekulové dynamiky vytvořen nový stav systému, který je následně vyhodnocen pomocí Monte Carlo přístupu. V nedávné době bylo v českém prostředí publikováno několik prací založených na této metodě (Nierostek 2021 [12], Nová 2022 [13]).

1.3 Přístup Alpha Foldu a jeho význam

Pro další vysvětlení významu této práce stručně popíšu přístup AlphaFoldu a také to, proč je i přes jeho nedávné úspěchy stále potřebná práce pro vylepšování *de novo* přístupů.

Programy AlphaFold a především AlphaFold 2 od společnosti Deep Mind dosáhly v oblasti odhadování konformace neznámých proteinů velmi dobrých výsledků. V soutěži CASP (13, 14) získaly první místa a konkrétně AlphaFold 2 v CASP 14 odhadl v drtivé většině případů struktury proteinů s RMSD⁵ menší než 2 Å.[14] To je bezkonkurenční výsledek, nicméně AlphaFold neodhaluje celý proces foldingu proteinu do 3D struktury, protože ji odhaduje přímo [15].

Na rozdíl od AlphaFoldu metody *de novo* zkoumají přímo proces foldingu. Je u nich totiž kladen důraz na fyzikální pozadí procesu skládání proteinů. Podstatným aspektem *de novo* metod je správné nastavení potenciálových funkcí. Cílem této práce bylo vytvořit program pro testování potenciálových funkcí.

1.4 Cíle práce

Cílem této práce je přispět k výzkumu proteinů vytvořením simulačního programu pro výzkum potenciálových funkcí. Program bude využívat zjednodušení peptidového vlákna na aminokyseliny reprezentované jako koule se středem v C_α uhlících (coarse-grained) a konstantním poloměrem. Jako návrh stavu je využito náhodné otočení části řetězce okolo pivota a Monte Carlo vyhodnocení návrhu stavu. Program dále využije Lennard-Jonesův potenciál jako funkci pro nezávislý potenciál a empiricky zjištěné funkce pro úhlové (bending a dihedrální) potenciály. Jedna z hlavních motivací pro vytvoření tohoto programu je umožnit testování různých potenciálových funkcí. Program má být také snadno uživatelsky přístupný pro člověka zvyklého pracovat s příkazovou řádkou a dostatečně jednoduchý, aby bylo možné jej dále vylepšovat a snadno upravovat i pro další účely.

⁵”Root Mean Square Deviation”, viz vlastní kapitolu.

Kapitola 2

Metodika

Program využívá Monte Carlo simulace s návrhem stavu pomocí pivota. Jako model bylo zvoleno zhrubení na úroveň jednotlivých aminokyselin. Ty jsou reprezentovány jako koule se středem v C_α atomech.

2.1 Stručný popis algoritmu simulace

Simulace probíhala ve třech fázích: inicializace, běh, finalizace.

2.1.1 Inicializace

Program pomocí vstupních argumentů nahraje parametry simulace a sekvenci peptidu, vytvoří soubory, do kterých budou později uložena simulační data a uspořádá aminokyseliny peptidu do rovné čáry ("rod" konformace).

2.1.2 Běh simulace

Pro zadaný počet cyklů se v rámci běhu simulace provedou následující kroky:

1. vypočítání potenciálu v nově navrženém stavu systému jako součet jednotlivých potenciálů
2. vypočítání Boltzmannova faktoru pro rozdíl mezi potenciály starého a nového systému
3. krok Monte Carlo pro přijetí/odmítnutí nového stavu pomocí náhody
4. samotné uložení nového stavu v případě jeho přijetí, případně jeho zamítnutí
5. zápsání veličin stávajícího stavu do výstupních souborů

2.1.3 Finalizace

Po doběhnutí všech kroků jsou spočítány průměry jednotlivých veličin a jejich chyby pomocí blokové metody. Část dat je pro tyto kroky přeskočena (přesný počet přeskočených kroků je uveden ve vstupním souboru pro danou simulaci, parametr `Skipped-Cycles`).

Podrobněji se budu jednotlivým krokům věnovat v dalších kapitolách.

2.2 Zhrubené modely

Pro simulace skládání proteinů se obecně využívá několik typů zjednodušení pomocí tzv. zhrubených modelů. Určité zjednodušení je z praktických důvodů nutné. Počet možných konformací proteinu je příliš velký na to, aby byl doporučena celý prozkoumán v reálném čase.

Pokud jde o prostor, ve kterém jsou simulace prováděny, jsou využívány mřížkové modely (v literatuře označován jako "lattice model") a pak spojitý prostor ("off-lattice model"). V případě mřížek jsou aminokyseliny reprezentovány jako kuličky, které mohou být pouze na průsečících přímek tvořících pravidelnou pravoúhlou 2D nebo 3D mřížku. Spojitý 3D prostor nabízí více konformací než 3D mřížka. Proto jsou simulace ve spojitém prostoru sice složitější a pomalejší, ale více odpovídají reálným podmírkám.¹

Ve spojitém 3D prostoru může být peptid reprezentován opět několika způsoby. Nejvyšší možné rozlišení, které dává smysl v kontextu prostorové struktury proteinu je atomární rozlišení ("all-atom resolution"). Dalšími jsou různé možnosti zjednodušení jednotlivých aminokyselin na jednu nebo několik kuliček.²

Například Postic 2021 [17] srovnává následující modely zjednodušení:

- pouze C_α atomy,
- pouze C_β atomy,
- C_α a C_β ,
- základní struktura proteinu (backbone),
- backbone peptidu a C_β atomy

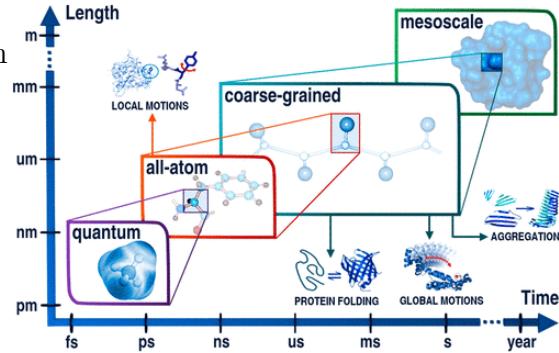
a další, viz obrázek 2.2. Každý ze způsobů zjednodušení má své výhody, nevýhody a specifika.

2.2.1 Model použitý v této práci

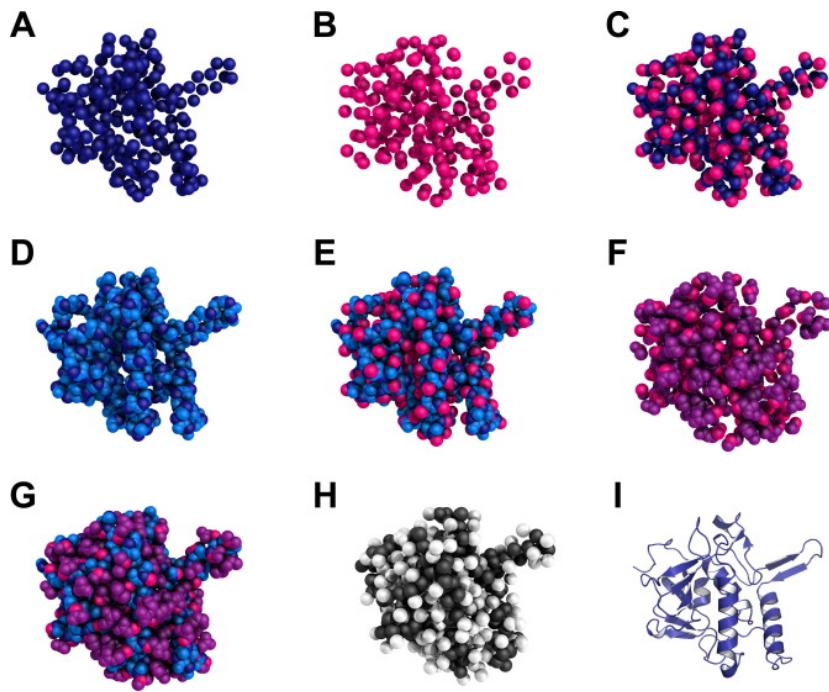
V této práci je využito zjednodušení na C_α atomy. Každá aminokyselina z proteinu je tedy reprezentována středem C_α atomu a velikostí aminokyseliny. Poloměr aminokyseliny byl standardně nastavena na 5 Å, což odpovídá nejbližší

¹Kmiecik 2016 [16].

²Kmiecik 2016 [16], Postic 2021 [17]



Obrázek 2.1: Zde jsou zobrazeny schematicky aplikační rozsahy pro různá rozlišení: kvantové, celoatomární, zhrubené a "mesoscale" rozlišení. Na obrázku jsou vidět přibližné rozsahy časových a prostorových jednotek používaných na dané úrovni zjednodušení. Obrázek pochází z práce Kmiecik 2016 [16].



Obrázek 2.2: Na obrázku vidíme více typů zjednodušení. (A) je zjednodušení na C_α atomy, (B) C_β , (C) $C_\alpha + C_\beta$, (D) backbone, tedy peptid bez postranních řetězců, (E) backbone + C_β , (F) pouze postranní řetězce, (G) všechny atomy, (H) model "MARTINI", který má reprezentace jak backbone tak postranních řetězců, tento model je více popsán v práci Marrink 2007 [18], (I) zjednodušená reprezentace celé struktury, používaná pro zdůraznění sekundárních struktur proteinů. Obrázek převzatý z práce Postic 2021 [17].

vzdálenosti dvou nesousedících aminokyselin. Do této vzdálenosti jsou zahrnutý i postranní řetězce aminokyselin. Tuto velikost je v rámci uživatelského nastavení programu možné měnit. Další vývoj programu počítá s možností, že velikost reprezentace aminokyseliny bude specifická pro jednotlivé typy aminokyselin. Délka vazeb mezi C_α uhlíky je 3.81 Å, což odpovídá jejich skutečné vzdálenosti (pro dva uhlíky C_α jdoucí za sebou v řetězci má vzdálenost velmi úzkou distribuci).³

2.3 Potenciály

Vhodné nastavení potenciálových funkcí je nezbytné pro dobře fungující program. Mají význam při kroku Monte Carlo, kdy jsou porovnány potenciály starého a nového systému a na základě toho je nový stav přijat nebo odmítnut.

V této práci jsou použity tři typy potenciálů: nevazebný, bending a dihedrální potenciál. Vazebný potenciál nebyl použit kvůli úzkým distribucím vzdáleností mezi sousedními aminokyselinami.

³Pavlíková 2022 [19]

2.3.1 Nevazebný potenciál

Nevazebný potenciál se počítá na základě vzdáleností daných reprezentací aminokyselin. Je více možností, jakou zvolit potenciálovou funkci. V této práci byl použit N. Metropolisem vybraný Lennard-Jonesův potenciál [8].

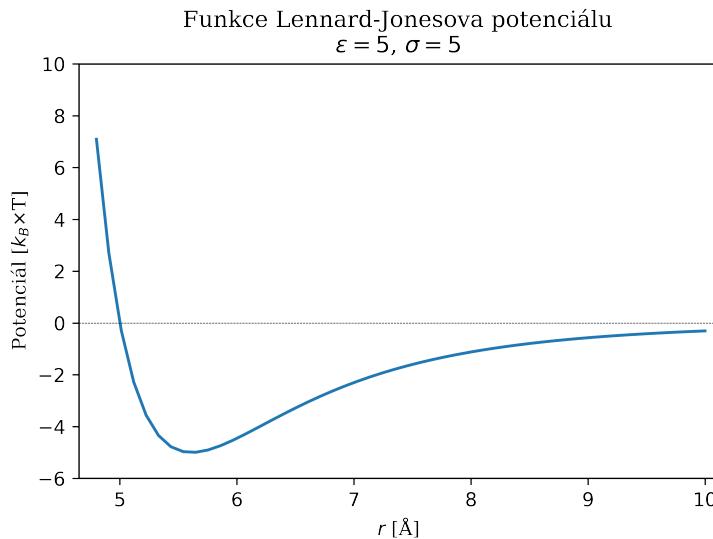
Lennard-Jonesův potenciál

Bud'te

- x_1, x_2 reprezentace částic
- $\epsilon(x_1, x_2)$ hloubka potenciálové jámy pro dané reprezentace částic,
- $r(x_1, x_2)$ vzdálenost mezi reprezentacemi částic a
- $\sigma(x_1, x_2)$ průměr poloměrů reprezentací částic (jsou to koule).

Pak Lennard-Jonesův nevazebný potenciál se spočítá takto:

$$U_{LJ}(x_1, x_2) = 4\epsilon(x_1, x_2) \times \left[\left(\frac{\sigma(x_1, x_2)}{r(x_1, x_2)} \right)^{12} - \left(\frac{\sigma(x_1, x_2)}{r(x_1, x_2)} \right)^6 \right] \quad (2.1)$$



Obrázek 2.3: Průběh funkce Lennard-Jonesova potenciálu pro zadané parametry.

Funkce nevazebného potenciálu přes všechny kombinace reprezentace aminokyselin v řetězci se pak spočítá takto⁴:

$$U_{LJ-all} = \sum_{i=1}^{n-2} \sum_{j=i+2}^n U_{LJ}(x_i, x_j) \quad (2.2)$$

⁴Jsou vynechány přímo sousedící reprezentace aminokyselin v řetězci, protože vzdálenosti sousedních kuliček se v rámci systému v této práci nemění, viz kapitolu o návrhu stavu.

Volba parametrů pro výpočet nevazebného potenciálu

V této práci byly poloměry reprezentací stanoveny jako 5 Å pro všechny typy aminokyselin.⁵ Hodnoty ϵ jsou v této práci specifické pro každou kombinaci typů aminokyselin. Konkrétní hodnoty ϵ byly převzaty z práce Tanaky 1976 [20], kde jsou uvedeny v tabulce pro energie kontaktu. Hodnoty byly před použitím převedeny na jednotky použité v této práci.

Úprava hodnot epsilon pro účely této práce

Tanaka ve své práci vypočítává energie interakcí mezi jednotlivými typy aminokyselin. Aby tyto hodnoty⁶ bylo možné použít jako parametr ϵ při výpočtu Lennard Jonesova potenciálu, bylo nutné je převést do simulačních jednotek.

Tanaka uvádí hodnoty v kcal × mol⁻¹ a jednotka energie v simulaci je $k_B \times 300\text{K}$.

Pro x vyjadřující číselně koeficient pro přenásobení hodnot v tabulce, $k_B = 1,380649 \times 10^{-23} \text{JK}^{-1}$ je Boltzmannova konstanta a $N_A = 6,022 \times 10^{23}$ je Avogadrovo číslo. Pak platí, že:

$$\begin{aligned} x \times k_B \times 300\text{K} &= \text{kcal} \times \text{mol}^{-1} \\ x &= \frac{\text{kcal}}{k_B \times 300\text{K} \times \text{mol}} \\ &= \frac{4184}{1,380649 \times 10^{-23} \times 300 \times 6,022 \times 10^{23}} \times \frac{\text{J}}{\text{J} \times \text{K}^{-1} \times \text{K}} \\ &= \frac{4184}{2494,2804834} \\ &\approx 1,677 \end{aligned} \tag{2.3}$$

Tedy jedna jednotka v tabulce Tanaky je 1,677 simulační jednotky energie.

Alternativní potenciálová funkce - Mieův potenciál

Pro výpočet nevazebného potenciálu se v molekulových simulacích používá i tzv. Mieův potenciál (Mie potential), který je zobecněním Lennard-Jonesova potenciálu. Pro stejné parametry, jako u Lennard-Jonesova potenciálu, vypadá rovnice pro Mieův potenciál takto (Werth 2017[21]):

$$U_{Mie}(x_1, x_2) = c\epsilon \times \left[\left(\frac{\sigma(x_1, x_2)}{r(x_1, x_2)} \right)^a - \left(\frac{\sigma(x_1, x_2)}{r(x_1, x_2)} \right)^b \right] \tag{2.4}$$

Kde konstanty a, b, c jsou zvoleny podle potřeby. Lennard-Jonesův potenciál by pak byl speciální případ Mieova potenciálu s $a = 12$, $b = 6$, $c = 4$. Mieova potenciálu může být při vhodném nastavení konstant a, b, c může být průběh funkce realističtější.

⁵Více popsáno v kapitole o modelu simulace.

⁶Viz Tanaka 1976, tabulka č. 3 [20]

2.3.2 Vazebný potenciál

Vazebný potenciál má za účel zohlednit délku vazeb mezi jednotlivými reprezentacemi aminokyselin při vyhodnocování pravděpodobnosti stavu při Monte Carlo kroku. Jako funkci je možné použít např. následující funkci hyperboly:

$$U_{bond}(i) = k(l_0 - l_i)^2 \quad (2.5)$$

kde k je konstanta, l_0 je daná střední délka vazby (např. vzdálenost mezi C_α atomy, 3.81Å) a l_i je délka i -té vazby v řetězci.

Proč nebyl použit?

Vazebný potenciál nebyl použit, protože délka vazeb v řetězci zůstává po celou dobu simulace vzhledem k použité metodě návrhu stavu konstantní. I vazebný potenciál by proto byl konstantní přes všechny stavy systému a vzhledem k metodě přijímání stavů založené na rozdílu potenciálů dvou stavů by neměl v simulaci význam.

Vazebný potenciál také nebyl použit, protože distribuce délek vazeb je velmi úzká [19].

Jeho odstraněním z počítačové simulace bylo ušetřeno (pro n kuliček v řetězci a c cyklů) $n \times c$ výpočtů.

2.3.3 Bending potenciál

Bending potenciál v této práci popisuje úhel mezi třemi po sobě jdoucími C_α atomy v řetězci. Jako funkce bending potenciálu byla během vývoje programu použita parabola, nicméně v hotové práci už je využívána funkce vytvořená empiricky z již známých struktur proteinů pomocí metody Boltzmannovy inverze [19].

Funkce paraboly vypadá pro 3 po sobě jdoucí reprezentace aminokyselin v řetězci, respektive úhel α reprezentující úhel mezi jejich středy a konstantu označující hodnotu úhlu ve vrcholu paraboly α_0 takto:⁷

$$U_{bend-parabole}(\alpha) = k \times (\alpha_0 - \alpha)^2 \quad (2.6)$$

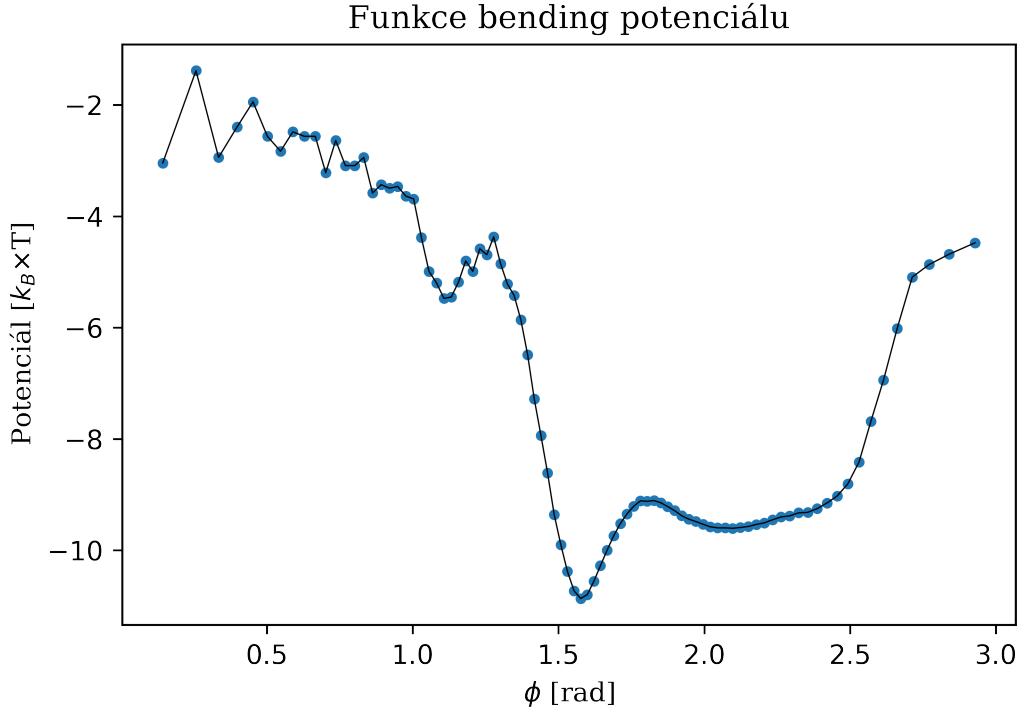
Úhel α nabývá hodnot $[0, \pi]$. Pro celý řetězec se bending potenciál spočítá takto:

$$U_{bend-all} = \sum_{i=1}^{n-2} U_{bend}(\alpha_{b_i, b_{i+1}, b_{i+2}}) \quad (2.7)$$

kde

- n je délka řetězce,
- b_i je střed reprezentace i -té aminokyseliny,
- α funkce úhlu mezi třemi body a
- U_{bend} je parciální funkce pro výpočet bending potenciálu (například zmíněná parabola).

⁷Tato funkce je navržena v publikaci Kolafa 2015[22].



Obrázek 2.4: Funkce bending potenciálu vytvořená pomocí Boltzmannovy inverze Markétou Pavlíkovou v rámci její bakalářské práce [19]. Tato potenciálová funkce byla použita k testování simulačního programu.

2.3.4 Dihedrální potenciál

Dihedrální potenciál popisuje úhel mezi polorovinami, viz obrázek 2.5. Pro čtyři po sobě jdoucí souřadnice středů reprezentací C_α atomů v řetězci b_1, b_2, b_3, b_4 jde o polorovinami $\overline{b_1b_2b_3}$ a $\overline{b_2b_3b_4}$ definovaný dihedrální úhel ϕ . Protože jde o úhel mezi polorovinami a nikoli rovinami, je funkce potenciálu definována v rozsahu $\phi \in [-\pi, \pi]$.

Výpočet dihedrálního úhlu je v simulačním programu implementován pomocí funkce `atan2`. Funguje podobně jako funkce `atan` ($\text{atan2}(y,x) = \text{atan}(y/x)$, pro $x > 0$), nicméně na rozdíl od funkce `atan` je možné ji využít pro celý interval $[-\pi, \pi]$. Celá rovnice pro výpočet dihedrálního úhlu ϕ pak vypadá takto:

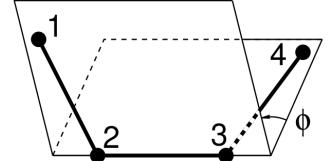
$$\phi = \text{atan2}(|v_2|v_1 \cdot v_2 \times v_3, (v_1 \times v_2) \cdot (v_2 \times v_3)) \quad (2.8)$$

Při vývoji programu byla opět použita pro funkci dihedrálního potenciálu parabola:

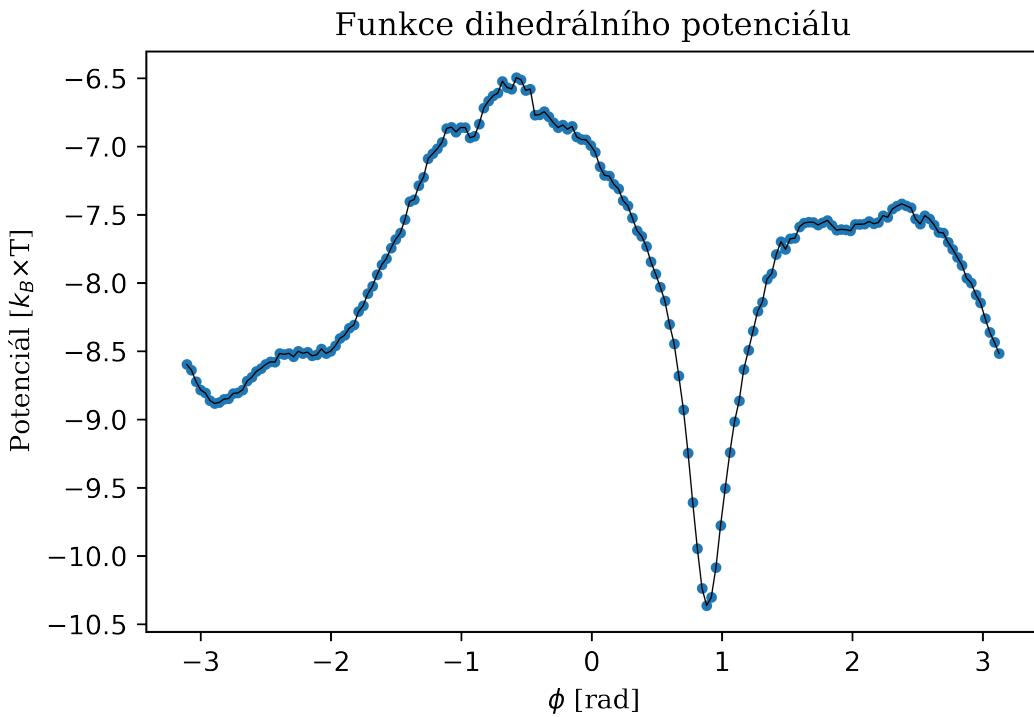
$$U_{\text{dihedral-parabola}}(\phi) = k \times (\phi_0 - \phi)^2 \quad (2.9)$$

kde ϕ_0 je hodnota úhlu ve vrcholu paraboly.

Hotový program využívá stejně jako pro bending potenciál empirickou funkci získanou ze známých struktur Boltzmannovou inverzí.



Obrázek 2.5: Na schématu jsou vidět dvě poloroviny dané body 1-4. ϕ označuje dihedrální úhel. Obrázek je převzatý z publikace Kolafa 2015 [22].



Obrázek 2.6: Graf ukazuje průběh funkce dihedrálního potenciálu použité při testování programu. Funkce byla získána pomocí Boltzmannovy inverze [19] z reálných struktur proteinů. Jsou zde vidět dvě výrazná lokální minima, jedno (globální) v bodě [0.88, -10.36] a druhé v bodě [-2.89, -8.88]. Oblasti kolem těchto minim odpovídají sekundárním strukturám alpha helixům a beta listům.

2.3.5 Boltzmannova inverze

Boltzmannova inverze je metoda pro získání potenciálové funkce při znalosti konformací reálných proteinů. Pro dataset proteinů se známou konfigurací jsou spočteny vzdálenosti a úhly mezi atomy těchto peptidů, tedy distribuční funkce. Pomocí Boltzmannovy inverze je získána funkce efektivního potenciálu (U_{eff}). Tento postup je popsán například v práci Chudoba a kol. 2017 [23].

Bud'

- k_B Boltzmannova konstanta,
- T teplota,
- g distribuční funkce a
- x úhel či vzdálenost (záleží o jaký typ potenciálu se jedná).

Pak efektivní potenciál pro daný úhel nebo vzdálenost pomocí Boltzmannovy inverze má tuto podobu:

$$U_{eff}(x) = -k_B T \log g(x) \quad (2.10)$$

2.4 Návrh stavu

Simulace probíhá tak, že se navrhují nové stavы systému a ty jsou následně přijaty, nebo odmítnuty. Existuje více možností navržení nových stavů simulovaného systému. Pro návrh stavu s následným vyhodnocením metodou Monte Carlo se používá molekulová dynamika⁸, či metody pracující s náhodou (např. návrh pomocí pivota).

Molekulová dynamika vytváří nové stavы na základě silových polí, tedy derivacích potenciálů a z toho vznikajícího pohybu. Je poměrně náročná z hlediska času,⁹ ale nový stav vytváří nenáhodně. Nové stavы molekulové dynamiky jsou zpravidla přijímány automaticky bez dalšího vyhodnocovacího kroku.

Metody, které pracují s náhodou vycházejí stejně jako molekulová dynamika z aktuálního stavu systému. Na rozdíl od molekulové dynamiky ale při vytváření nového stavu příliš nereflektují potenciály v systému a nový stav vytvoří náhodně podle daných pravidel. Potenciál v systému začíná hrát roli až při vyhodnocování nového stavu.

Vyhodnocování stavu musí reflektovat, jestli se celkový potenciál zvýšil, nebo snížil. Na základě toho je přijat, nebo odmítnut nový stav.

2.4.1 Návrh stavu pomocí pivota

V této práci byla pro vytváření nových stavů použita metoda otočení řetězce okolo pivota. Jeho konkrétní implementace je taková, že algoritmus

1. náhodně vybere pivot - jednu z reprezentací aminokyselin v řetězci,
2. náhodně vybere směr k C, či N konci řetězce,
3. náhodně vytvoří jednotkový vektor,
4. náhodně vybere úhel v určitém rozmezí (bylo použit interval [0,0.3]rad, horní hranice je definována ve vstupním souboru programu),
5. vytvoří matici otočení pro daný vektor a úhel,
6. souřadnice všech reprezentací aminokyselin v celé náhodně vybrané části řetězce ve směru od pivota transformuje pomocí matice otočení.

Tento algoritmus je poměrně rychlý. Po vytvoření matice otočení proběhne už jen 9 násobení a 6 součtů nejvýše pro každou reprezentaci aminokyseliny v peptidu, což je složitost $O(n)$ pro n aminokyselin v peptidu.

⁸Tzv. Hamiltoniánské Monte Carlo, srov. práce Nová 2022 [13] a Nierostek 2021 [12]

⁹Náročnost ve smyslu "time cost".

Limity návrhu pomocí pivotu

Princip návrhu stavu pomocí pivot moves byl poměrně používaný pro simulace na mřížce (lattice), ale dnes je daleko hojněji používána molekulová dynamika. Pivot moves jsou totiž poměrně problematické pro simulace delších řetězců¹⁰ a rozvětvených molekul¹¹.

Matice otočení

Matice otočení (A) využitá algoritmem Pivot moves má pro vektor n a úhel α tuto podobu (jde o 3D prostor):

$$A = \begin{pmatrix} \cos \alpha + n_1^2(1 - \cos \alpha) & n_1 n_2(1 - \cos \alpha) - n_3 \sin \alpha & n_1 n_3(1 - \cos \alpha) + n_2 \sin \alpha \\ n_1 n_2(1 - \cos \alpha) + n_3 \sin \alpha & \cos \alpha + n_2^2(1 - \cos \alpha) & n_2 n_3(1 - \cos \alpha) - n_1 \sin \alpha \\ n_1 n_3(1 - \cos \alpha) - n_2 \sin \alpha & n_2 n_3(1 - \cos \alpha) + n_1 \sin \alpha & \cos \alpha + n_3^2(1 - \cos \alpha) \end{pmatrix} \quad (2.11)$$

Matice byla převzata z Wikipedie, heslo "Otočení" [24].

2.5 Monte Carlo vyhodnocení stavu

Princip simulací s Monte Carlo byl poprvé popsán Nicholasem Metropolisem v roce 1953 [8]. Ten použil Monte Carlo přístup pro simulaci systému tuhých kuliček s nevazebným Lennard-Jonesovým potenciálem a ukázal, že jeho použití je legitimní. Použití tohoto typu vyhodnocení stavu umožňuje nastavit volněji návrh stavu s tím, že je právě při Monte Carlo kroku nepravděpodobný stav zamítnut. Monte Carlo navíc a priori nevylučuje žádný možný stav systému z prostoru všech stavů, ale pouze s vysokou pravděpodobností nepřijme nový nepravděpodobný stav.

Pro Monte Carlo je potřeba přiřadit novému stavu určitou pravděpodobnost. Po vytvoření nového stavu je proto spočítán potenciál systému. Na základě rozdílu mezi potenciálem starého a nového stavu je spočítán Boltzmannův faktor, který dává pravděpodobnost, se kterou má být nový stav přijat.

Je také vygenerováno náhodné číslo v intervalu (0,1). Může dojít ke třem situacím:

1. Nový stav má nižší potenciál než starý stav, je rovnou přijat.
2. Nový stav má vyšší potenciál než starý stav,
 - (a) náhodné číslo je nižší než Boltzmannův faktor, nový stav je přijat nebo
 - (b) náhodné číslo je vyšší než Boltzmannův faktor, nový stav je odmítnut.

¹⁰Problematičnost simulování delších řetězců se projevila i ve výsledcích této práce, viz kapitola výsledků a závěrů práce, kde navrhoji i možnosti zlepšení návrhu stavu pro delší řetězce.

¹¹Jako příklad je možné uvést situaci, kdy molekula má v sobě cyklus. Pak výše popsaný algoritmus v jeho základní podobě není možné vůbec použít.

2.5.1 Boltzmannův faktor

Pomocí Boltzmannova faktoru je interpretován rozdíl v potenciálu dvou systémů za dané teploty jako pravděpodobnost [8].

Pro

- teplotu T ,
- potenciál starého stavu U_{old} ,
- potenciál nového stavu U_{new} a
- Boltzmannova konstantu k_B ,

vypadá rovnice Boltzmannova faktoru takto¹²:

$$f_{Boltzmann} = e^{(U_{old} - U_{new})/(k_B \times T)} \quad (2.12)$$

2.5.2 Simulované žíhání

Simulované žíhání (také simulated annealing) je technika používaná při simulacích, v rámci níž se postupně snižuje teplota v systému. Efekt této techniky je, že se zprvu přijímají i stavy, které by byly za běžných okolností méně pravděpodobné. Systém má díky tomu větší dynamiku (a nezůstane v lokálním minimu). Po snížení teploty už se přijímají pouze stavy, které mají relativně menší nárůst potenciálu oproti předchozímu stavu. Systém se tak více stabilizuje.¹³

V této práci byla technika simulovaného žíhání implementována tak, že uživatel dá programu jako parametry teplotu na počátku (T_{init}) a teplotu na konci (T_{final}). Pro i -tý cyklus z celkového počtu n cyklů je pak teplota systému T počítána jako:

$$T = T_{init} + \frac{(T_{final} - T_{init}) \times i}{n} \quad (2.13)$$

2.6 Simulační program

Pro implementaci navržené simulace byl vytvořen vlastní simulační program. V rámci implementace bylo dbáno na to, aby, kromě správnosti provádění všech kroků simulace, byl program i efektivní a přístupný pro uživatele. V následujících podkapitolách jsou popsány základní vlastnosti tohoto programu. Celý program včetně uživatelské a technické dokumentace je možné najít na stránkách projektu na GitHubu.¹⁴

¹²Metropolis 1953 [8].

¹³Více je technika popsána například v práci Zhang 2020 [25]. Metoda byla použita například i v práci [12].

¹⁴Projekt se jmenuje "PivotMoveSimulation", url projektu je "<https://github.com/ForgoRew/PivotMoveSimulation>"

2.6.1 Programovací jazyk

Jako programovací jazyk byla zvolena Java 17. Jejími přednostmi je

- relativně vysoká rychlosť (například v porovnání s jazykem Python),
- vysoká portabilita (Java je staticky kompilována a spouštěna pomocí Java Virtual Machine, která je k dispozici pro operační systémy Windows, Linux i MacOS),
- její široké rozšíření a znalost i mezi méně zkušenými programátory,
- dobrá čitelnost kódu.

Zvláště poslední tři vlastnosti Javy byly důležité, aby program mohl být v budoucnu dál využíván a vyvýjen.

2.6.2 Datové struktury, třídy a metody

Vlastní program tvoří následující třídy (v abecedním pořadí), v dalších podkapitolách se každé budu věnovat podrobně:

1. App
2. Ball
3. DataRange
4. MyWatches
5. MyWriter
6. Physics
7. SimRunVars
8. SimSpace
9. StepVars

App

Z této třídy je program spouštěn (standardně pomocí metody `main`) a obsahuje metodu pro samotný běh simulace (`pivotMovesSimulation`), metody pro zápis informací o průběhu simulace do souborů (`countAndNoteAll`, `notePositions`, `noteDistances`) a metody pro vyhodnocení dat simulace ve finalizační fázi simulace (`countErrBlockMethod`, `writeLog`).

Ball

Objekty této třídy reprezentovaly jednotlivé aminokyseliny v řetězci. Obsahují atributy, které určují jejich pozici (`coordinates`), typ (`type` a `code`) a velikost (`size`, `weight`).

DataRange

Objekty této třídy jsou využívány při výpočtu chyby blokovou metodou a reprezentují určitý úsek dat.

MyWatches

Objekt této třídy jednoduchým způsobem spočítá čas simulace. Zaznamená čas začátku simulace a její konec a vypočítá rozdíl mezi nimi (v ms).

MyWriter

Objekt této třídy zjednodušuje používání `BufferedWriteru`, který je jedním z jeho atributů. Pomocí metody `writeLine` přidá řetězec, který dostane jako argument, do výstupního souboru. Metoda `writeAll` pak do výstupního souboru přidá celý obsah vstupního souboru, který dostane (otevřený) jako argument.

Physics

Tato třída je v podstatě knihovnou fyzikálních a matematických funkcí pro běh programu. Obsahuje

- metody pro výpočet různých typů potenciálů, např.
 - Lennard-Jones potenciál (`lennardJonesPotential`)
 - další potenciály vhodné pro vývoj a testování simulace, např. `squareWellPotential`,
 - tříčásticové a čtyřčásticové potenciály (`bendingPotential`, `dihedralPotential`)
- metody pro výpočet síly (využity při vývoji programu),
- metody pro práci s vektory a celkově vytváření nového stavu systému, např.
 - skalární součin vektorů (`dot`) a další,
 - vytvoření a použití matice rotace (`makeMatrixOfRotation`, `moveBallsByMatrix`),
- výpočet Boltzmannova faktoru (`boltzmannsFactor`), či
- získávání náhodných hodnot (náhodného vektoru, či úhlu, `getRandomNormalizedVector`, `getRandomAngle`).

SimRunVars

Objekty této třídy v sobě obsahují některé atributy pro aktuálně probíhající simulaci. Je vždy vytvořen jeden objekt této třídy pro daný běh simulace a obsahuje

- počet přijatých/odmítnutých stavů (`accepted`, `rejected`),

- číslo aktuálního cyklu simulace,
- sumy jednotlivých veličin (např. `sumPotential`)
- průměry některých veličin, které jsou počítány po skončení běhu simulace (např. `avgPotential`).

Také obsahuje metodu pro výpočet průměrů (`makeAvgs`) a více typů konstruktorů (kvůli testování).

SimSpace

Objekt třídy `SimSpace` obsahuje všechny vstupní parametry simulace a vstupní a výstupní soubory. Pomocí něj jsou metodám předávány parametry simulace. Obsahuje také aktuální stav reprezentací aminokyselin (pole `balls`). Jeho součástí je také metoda pro vytvoření `tcl` skriptu, který slouží ke správnému zobrazení stavů systému v programu VMD ("Visual Molecular Dynamics") [26].

Třída `SimSpace` obsahuje metody pro iniciaci simulace. Samotný konstruktor této třídy přijímá název simulace jako řetězec znaků a díky němu nahraje vstupní soubor, parametry simulace, tabulkou s parametry epsilon pro nevazebný potenciál. Dále vytvoří kódování typů aminokyselin a iniciuje vznik pole reprezentací aminokyselin.

StepVars

Objekty této třídy obsahují hodnoty specifické pro jeden krok simulace. Jsou v něm uložené hodnoty jako

- index pivota (`pivotIndex`),
- potenciál systému (`potential`),
- gyrační poloměr (`Rg`) a další.

Kromě toho obsahuje ještě nově vytvořený stav systému, který ještě není přijatý (`newBalls`).

2.6.3 Vstupy a výstupy

Simulační program potřebuje následující vstupní soubory:

- vstupní soubor ve formátu `JSON`, ve kterém jsou uvedeny parametry simulace,
- soubor ve formátu `FASTA` se sekvencí simulovaného peptidu,
- `csv` tabulka s parametrem "epsilon" pro výpočet párového, nevazebného potenciálu a
- `csv` tabulky s hodnotami s hodnotami pro bending a dihedrální potenciál.

Program v průběhu svého běhu vytvoří následující soubory ve výstupní složce:

1. **log** soubor se stručným souhrnem důležitých informací o simulaci a jejím průběhu. Soubor obsahuje:
 - vstupní parametry,
 - průměry a chyby jednotlivých veličin,
 - zprávu o úspěšném provedení simulace/errory
2. **csv** soubor s důležitými číselnými daty pro každý krok. Ten zahrnuje:
 - pořadí daného kroku simulace,
 - boolean, zda byl vygenerovaný stav simulace přijat,
 - celkový potenciál v systému,
 - nevazebný potenciál,
 - bending potenciál,
 - dihedrální potenciál,
 - gyrační poloměr proteinu,
 - vzdálenost konců řetězce,
 - náhodnou pravděpodobnost z intervalu (0,1), která byla zvolena při Monte Carlo kroku simulace,
 - vypočítaný Boltzmannův faktor při vyhodnocování nově navrženého stavu konformace proteinu.
3. soubor s průměry hodnot některých veličin,
4. **xyz** soubor se souřadnicemi středů aminokyselin v jednotlivých krocích simulace,
5. **tcl** skript pro danou simulaci, který slouží k nastavení parametrů vizualizace peptidu v jednotlivých krocích v programu VMD.

2.7 Jednotky simulace

Pro fyzikálně relevantní fungování simulace je potřeba vytvořit systém jednotek, ve kterém simulace probíhá. Čas, délka, hmotnost a teplota jsou důležité veličiny pro chod simulace. Základní SI jednotky těchto veličin ale nejsou pro potřeby této simulace příliš vhodné.¹⁵ V rádu jednotek totiž popisují systémy na úrovni makrosvěta. Kromě těchto základních veličin bylo ještě potřeba určit jednotku energie v simulaci.

¹⁵V případě simulací jednotlivých proteinů jde o základní jednotky sekundy (s), metry (m), kilogramy (kg) a Kelviny (K).

2.7.1 Délka

Simulace probíhá na úrovni proteinů, a tak je potřeba nastavit vzdálenosti v rádu 10^{-10}m . Běžná vzdálenost mezi C_α atomy dvou aminokyselin spojených peptidickou vazbou je ale $3.81 \times 10^{-10}\text{m}$. V chemii byla kvůli častému využití tohoto rozměru délky zavedena vlastní jednotka Ångström, značená jako "Å", pojmenovaná po švédském fyzikovi stejného jména¹⁶. Protože $1\text{\AA} = 10^{-10}\text{m}$, je tato jednotka velmi vhodná a široce využívaná pro simulaci foldingu proteinů.

2.7.2 Hmotnost

Hmotnost byla zvolena jako jedna AMU (atomic mass unit), značená jako "u", protože 1u je definována jako $1/12$ hmotnosti uhlíku C_6^{12} bud.¹⁷ Hmotnost běžných proteinů bude tedy v rádu tisíců "u".

2.7.3 Teplota

Jako jednotka teploty byl zvolen Kelvin, základní jednotka systému SI. Je dostačující, protože simulace jsou myšlené tak, že probíhají přibližně okolo pokojové teploty,¹⁸ nebo mírně vyšší/nižší.

2.7.4 Boltzmannova konstanta

Pro další výklad je potřeba zmínit Boltzmannovu konstantu. Boltzmannova konstanta je podle redefinice jednotek SI z roku 2019 [27] jednou ze sedmi definujících konstant, ze kterých jsou odvozeny základní jednotky SI. Hodnota Boltzmannovy konstanty je určena jako $1.380649 \times 10^{-23}\text{JK}^{-1}$.

2.7.5 Energie

Jako jednotka energie při simulaci byla stanovena $k_B \times 300\text{K}$, tedy explicitně zapsáno

$$k_B \times 300\text{K} = 1.380649 \times 10^{-23} \times 300\text{J}$$

V těchto jednotkách se běžně pohybuje hodnota potenciálové funkce během simulace v rádu jednotek až desítek.

2.7.6 Čas

Délka, hmotnost, teplota a energie jsou ve vzájemném vztahu, proto nebylo možné čas určit podobně jako v případě hmotnosti, délky a teploty, ale bylo třeba

¹⁶Celým jménem Jonas Anders Ångström.

¹⁷1 AMU lze také zapsat jako $u = 1.66053906660(50) \times 10^{-27}\text{kg}$.

¹⁸ $26.85^\circ\text{C} = 300\text{K}$

ji stanovit na základě určených jednotek ostatních veličin.

$$\begin{aligned}
 [t] &= [l] \times \sqrt{\frac{[m]}{[E]}} \\
 &= \text{\AA} \times \sqrt{\frac{u}{k_B \times 300\text{K}}} \\
 &= 10^{-10}\text{m} \times \sqrt{\frac{1.66053906660(50) \times 10^{-27} \text{kg}}{1.380649 \times 10^{-23} \times 300\text{J}}} \\
 &\approx 1.094 \times 10^{-12}\text{s}
 \end{aligned} \tag{2.14}$$

Jednotky času ale nejsou během simulace, která byla obsahem této práce, relevantní, protože stavy jsou diskrétní a jednotlivé stavy nemají určený žádný čas.

Veličina	Jednotka	Vzorec
délka	Ångström	$[l] = 1\text{\AA} = 10^{-10}\text{m}$
hmotnost	AMU	$[m] = 1u = 1.66053906660(50) \times 10^{-27} \text{kg}$
teplota	Kelvin	K
energie	k_B , K	$[E] = k_B \times 300\text{K} = 1.380649 \times 10^{-23} \times 300\text{J}$
čas	vlastní	$[t] = [l] \times \sqrt{\frac{[m]}{[E]}} = \text{\AA} \times \sqrt{\frac{u}{k_B \times 300\text{K}}} \approx 1.094 \times 10^{-12}\text{s}$

Tabulka 2.1: Shrnutí volby simulačních jednotek.

2.8 Vyhodnocení simulace

Začátek simulace (ekvilibrace) nebyl při vyhodnocení simulace použit. Byla použita data až po určitém počtu kroků. Pro popis výsledků simulace byly použity následující hodnoty:

- průměr veličin měřených během simulace,
- chyby veličin, které byly počítány blokovou metodou a
- RMSD (root mean square deviation).

2.8.1 Průměry

Průměrné hodnoty veličin byly vypočítány běžným vzorcem pro aritmetický průměr přes všechny hodnoty. Do průměrů nebyly započítány hodnoty v cyklech definovaných uživatelem jako určené k přeskročení.

Průměr byl vypočítán pro následující veličiny:

- celkový potenciál,
- gyrační poloměr,
- vzdálenost konců řetězce,

2.8.2 Chyby

K výpočtu chyb byla použita bloková metoda, jak je popsána zde [28], protože je potřeba zohlednit možné autokorelace v simulaci. Bloková metoda funguje tak, že se

1. data rozdělí na 16 bloků,
2. v rámci těchto bloků se spočítají průměry, označme je a_1, a_2, \dots, a_{16} a
3. spočítá se průměr z těchto partikulárních průměrů, a_{all} .

Poté se chyba e spočítá následovně:

$$e = \sqrt{\sum_{i=1}^{16} (a_{all} - a_i)^2}$$

2.8.3 RMSD

Root Mean Square Deviation (RMSD) se používá pro hodnocení prostorové podobnosti dvou proteinových struktur. Na základě této hodnoty se usuzuje, jak byla simulace struktury proteinu kvalitní. RMSD používá pro hodnocení výsledků simulací i soutěž CASP14¹⁹.

V rámci této práce byl pro výpočet RMSD použit program VMD [26].

¹⁹Srov. výsledky soutěže CASP14 [14].

Kapitola 3

Výsledky a diskuze

V této kapitole jsou popsány výsledky práce, program a jeho testování na reálných proteinech.

3.1 Simulační program

Simulační program funguje na základě principů popsaných v předchozích kapitolách. Je volně dostupný na GitHubu [29],¹ kde je umístěna i jeho uživatelská dokumentace (česky)² a programátorská dokumentace (anglicky). Program je dostupný ve dvou variantách (se stejnou funkčností). První je zdrojový kód jazyka Java, spravovaný jako Maven projekt. Druhou možností je tzv. *jar* balíček,³ což je i doporučený způsob použití v uživatelské dokumentaci pro jeho jednoduchost.

3.2 Testování simulačního programu

Pro ověření funkčnosti programu bylo provedeno testování na reálných proteinech.

3.2.1 Výběr proteinů pro simulace

Proteiny byly vybrány náhodně z datasetu, který použila Markéta Pavlíková ve své ještě nepublikované práci pro stanovení hodnot úhlových a nevazebních potenciálů pomocí Boltzmannovy inverze. Konkrétně šlo o proteiny uvedené v tabulce 3.1.

Proteiny byly vybrané tak, aby konformace byla co nejvíce ovlivněna nevazebními interakcemi v rámci struktury proteinu a aby proteiny byly simulovatelné s využitím návrhu stavu pomocí pivota. Proto nebyly zahrnuty glykoproteiny, lipoproteiny, metaloproteiny, proteiny s S-S můstky, dlouhé řetězce (více než 300 aminokyselin) a proteiny s více než jedním peptidickým vláknem.

Fasta sekvence byly získány z databáze PDBe⁴. Sekvence jednotlivých proteinů uvádíme v přílohách.

¹Odkaz na stránky projektu na GitHubu zde: <https://github.com/ForgoRew/PivotMoveSimulation>.

²Jde o soubor "README.md".

³Java Archive, pro více informací viz oficiální specifikaci archivu jar [30].

⁴Protein Databank in Europe.

ID	počet AK v řetězci	RMSD [Å]
1dyw	173	18.89
1l23	164	23.62
2m1t	22	7.97
2msj	66	11.02
2nbr	173	55.25
2ruo	16	6.87
2yty	88	13.29
3lj8	143	23.48
4mq3	129	25.39
5ttt	111	23.52
5wrx	13	3.91
6iws	73	13.39

Tabulka 3.1: Přehled simulovaných proteinů - ID proteinu, délka řetězce (počet aminokyselin v řetězci) a RMSD.

3.2.2 Parametry zadané simulace

Pro n aminokyselin ve fasta sekvenci daného proteinu bylo provedeno $n \times 200000$ cyklů simulace. Polovina z těchto cyklů byla přeskočena jako ekvilibrační čas. Funkcí nevazebného potenciálu byla Lennard-Jonesova funkce s párově specifickými hodnotami epsilon pro dané dvojice aminokyselin. Velikost reprezentace aminokyseliny (hodnota σ u LJ potenciálu) byla zvolena na 5 Å. Pro úhlové bending a dihedrální potenciály byly zvoleny empirické potenciálové funkce. Nastavení potenciálových funkcí se nelišilo od základního nastavení simulace popsaného v kapitole Metodika.

Kromě tohoto základního nastavení byla ještě použita možnost simulovaného žíhání. Počáteční teplota byla nastavena na 500K a v průběhu simulace klesala lineárně až ke konečným 300K. Simulace probíhala na dvě fáze, kdy pokaždé bylo simulované žíhání provedené znovu. První polovina cyklů byla přeskočena a započítána až druhá polovina.

Data byla zapisována jednou za každých 100 cyklů. Jako maximální úhel otočení bylo zvoleno 0.5rad, což je poměrně vysoká hodnota. Této vysoké hodnotě odpovídá poměrně nízký podíl přijatých nových stavů u delších řetězců.⁵ Tím, že mohlo dojít k tak velkým změnám v jednom kroce, se mohly častěji vytvářet nepravděpodobné konfigurace. Těmi mohlo být například například protnutí reprezentací aminokyselin, nebo nepravděpodobný bending, či dihedrální úhel. Proto by bylo vhodné, aby byl při dalším použití programu zvolen maximální úhel otočení menší.

⁵Pro proteiny o délce řetězce alespoň 111 aminokyselin byl průměrný podíl přijatých stavů ku všem $\approx 0,024$. Oproti tomu u kratších peptidů byla tato hodnota ≈ 0.114 , tedy výrazně vyšší.

Například pro protein 1DYW vypadal vstupní soubor takto:

```
1dyw.json:  
{  
    "TypeOfSimulation": "Pivot-ChainMoves",  
    "TypeOfPotential": "Lennard-Jones",  
    "SimulationMatrixFileName": "AZ-Tanaka.csv",  
    "FASTAFileName": "1dyw.fasta",  
    "BallDiameter": 5,  
    "NumberOfCycles": 24800000,  
    "NumberOfSkippedCycles": 17400000,  
    "Temperature-Init": 500,  
    "Temperature-Final": 300,  
    "BendingPotentialTableName": "bendingPotential.csv",  
    "DihedralPotentialTableName": "dihedralPotential.csv",  
    "RotationRange": 0.5,  
    "LengthOfBond": 3.81,  
    "XYZFreq": 100,  
    "DATAFreq": 100,  
    "RenderTCL": false  
}
```

3.2.3 Výsledky simulací

Ukázky simulovaných proteinů

Z ukázky struktur simulovaných proteinů na obrázku 3.1 je vidět, že simulace delších řetězců (111 aminokyselin a delší) nevyšly ideálně. U některých struktur je zřejmá dlouhá smyčka vybíhající z jádra složeného proteinu. Konkrétně se jedná o 1dyw, 1l23, 2nbr, 3lj8, 4mq3 a 5ttt. Naopak zbývající kratsí proteiny (88 aminokyselin a méně) byly zřejmě simulované očekávaně, protože došlo k jejich sbalení.

Diskuze: proč nebyly delší řetězce dobře sbalené?

U delších řetězců (alespoň 111 aminokyselin v řetězci) zůstala prostřední část řetězce nesbalená a vyčnívala ze sbalené části proteinu ve formě jakési smyčky. Domnívám se, že to je způsobeno metodou návrhu stavů, tedy otočení řetězce okolo osy náhodného směru, procházející náhodným pivotem o náhodný úhel. Tento návrh stavu totiž umožňuje například změnu úhlu mezi třemi po sobě jdoucími reprezentacemi aminokyselin pouze, když je prostřední z nich vybrána jako pivot.

Pro představu uvedu příklad návrhu nového stavu u proteinu 1L23. Jedinou možností, jak ovlivnit úhly ve smyčce, která ze struktury vyčnívá, je vybrat jako pivota aminokyselinu z této smyčky. To je také jediná možnost, jak přidat tuto smyčku ke struktuře proteinů. Téměř jakýkoliv nově navržený stav pak ale nebude přijat. I malý vybraný úhel totiž udělá relativně velkou změnu pozic u vzdálenějších aminokyselin. V kompaktní sbalené části struktury pak dojde snadno ke změnám, které nejsou téměř přijatelné.

Tento závěr je v souladu s tím, že delší proteiny měly řádově nižší poměr přijetí nových stavů oproti kratším proteinům (2,4% oproti 11,4%). V závěrech navrhují vylepšení programu, které by tomuto jevu mohlo zabránit.

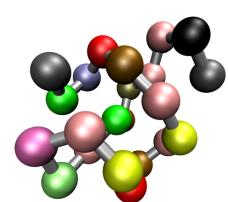
1DYW



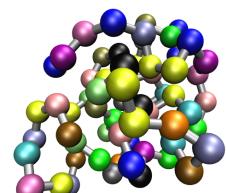
1L23



2M1T



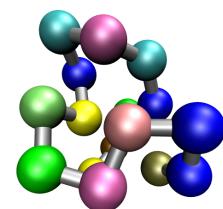
2MSJ



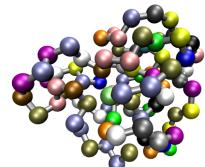
2NBR



2RUO



2YTY



3LJ8



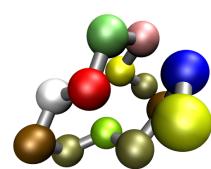
4MQ3



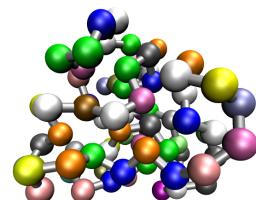
5TTT



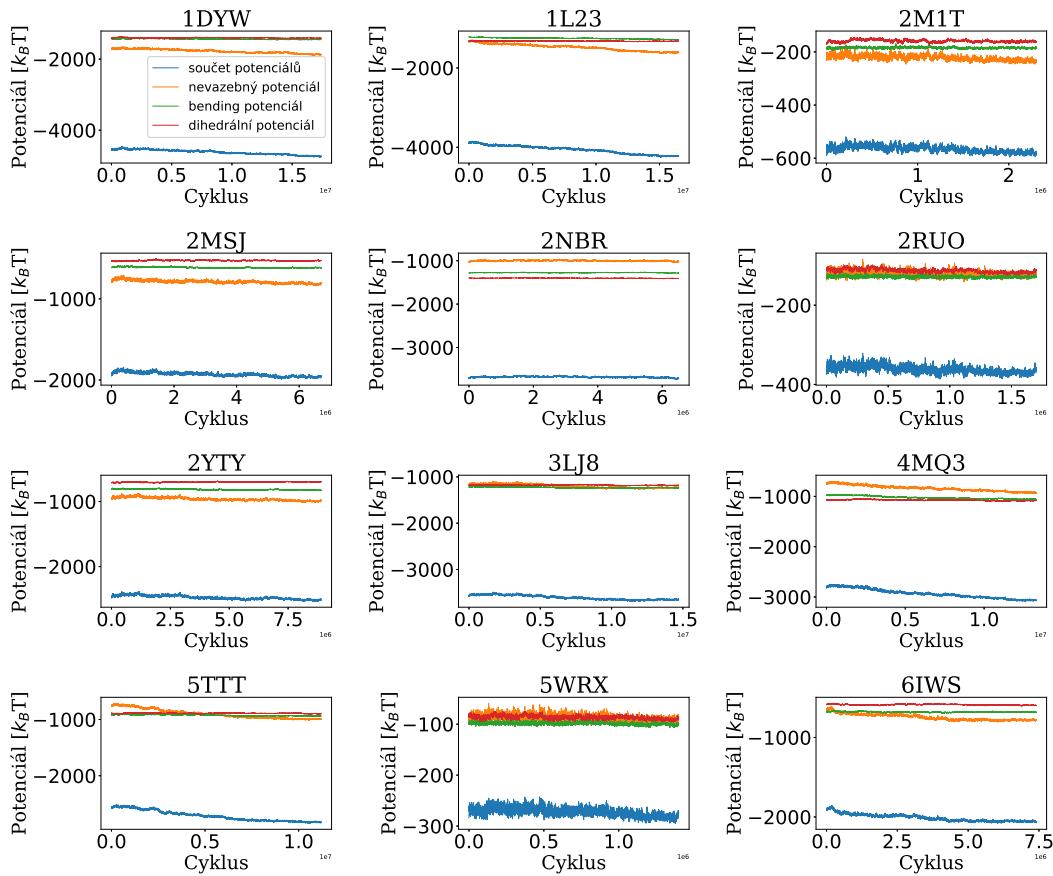
5WEX



6IWS



Obrázek 3.1: Ukázky struktur proteinů po skončení simulace. Barevný kód jednotlivých aminokyselin viz obrázek 3.3



Obrázek 3.2: Průběhy potenciálů simulovaných proteinů. Použita pouze data z produktivního části běhu simulace. Legenda u prvního grafu platí pro všechny. Potenciál se mění v důsledku změny teploty.



Obrázek 3.3: Barevný kód použitý v ukázkách simulovaných struktur.

Hodnoty potenciálů

Grafy vývoje potenciálových funkcí během simulace jsou vidět na obrázku 3.2. U části z nich je vidět mírný vzrůst potenciálu. To je možné interpretovat jako mírné rozvolnění struktury, která z prvního cyklu simulace již byla zchlazena na 300K. V druhém cyklu se opět zvedla na počátečních 500K a opět lineárně klesala ke konečným 300K. Na grafech je také vidět postupné snižování rozdílu mezi minimy a maximy pílků v potenciálech.

Také je možné si všimnout, že potenciál vyšel u většiny simulací záporně. To je očekávaný výsledek vzhledem k tomu, že hodnoty jednotlivých potenciálových funkcí jsou také záporné až na nevazebný, Lennard-Jonesův potenciál, který může vyjít kladný, pokud se protínají jednotlivé reprezentace aminokyselin. Záporný potenciál tedy ukazuje to, že k této nežádoucí situaci nedocházelo.

Potenciál hrál v této simulaci roli jen při vyhodnocování nově navrženého stavu během Monte Carlo kroku, kde ale byl důležitý rozdíl mezi potenciálem předchozího a nově navrženého stavu. Protože jsou potenciálové funkce stejné pro oba stavy, nemá případná k potenciálové funkci přičtená konstanta žádný vliv. Proto není jinak relevantní, zda je potenciál v systému kladný nebo záporný.

Podobnost simulovaných struktur s reálnými

Je možné si všimnout, že kratší proteiny jsou dobře sbalené, stejně jako ty reálné. To ukazuje i RMSD, které dosahuje u kratších proteinů (88 aminokyselin a méně) průměrné hodnoty 9,04Å.

Kapitola 4

Závěry

V této práci byl navržen a implementován model simulace proteinů se zhrubením na aminokyseliny reprezentované jejich středem (C_α atomy) a poloměrem (při testování byla zvolena hodnota poloměru pro všechny aminokyseliny na 5 Å). Vzniklý simulační program využívá metodu návrhu stavů pomocí otočení části řetězce kolem pivota a vyhodnocení stavu metodou Monte Carlo. Program také implementuje některé nástroje pro vyhodnocení simulace (výpočet průměru ze simulovaných veličin a výpočet chyby blokovou metodou). Architektura programu je vytvořena tak, aby umožňovala snadné změny v implementaci potenciálové funkce.

Testování a limity programu

Program byl testován na 12 proteinech a byl zjištěn rozdíl v kvalitě simulací pro krátké (méně než 88 aminokyselin) a dlouhé (111 aminokyselin a více) řetězce proteinů. Zatímco při simulacích krátkých řetězců se struktury sbalily podle očekávání, u dlouhých proteinů se nesbalila prostřední část řetězce a vytvořila vyčnívající smyčku. Tomu by se dalo zabránit rozšířením možností návrhu stavů. Vývoj potenciálu v systému neukázal nic nečekaného. V průběhu simulace má potenciál spíše klesající trend, který byl způsobený použitím metody simulovaného žíhání.

Možnosti vývoje algoritmu a programu

Na závěr ukazují další možnosti vývoje programu. Kromě pohybu řetězce okolo pivota je pro simulaci dlouhých řetězců nutné implementovat další typ návrhu stavu, který bude fungovat lokálně. Dále navrhoji drobná vylepšení a změnu verze programovacího jazyka na Java 8.

Přidání další metody návrhu nového stavu

Pro využití programu pro dlouhé řetězce je nutné rozšířit metodu návrhu stavu. Konkrétně je potřeba přidat možnost pohybu malého segmentu vlákna, například "svihadlový" pohyb. Ten by fungoval tak, že budou vybrány dva náhodné pivoty z reprezentací aminokyselin a celá podmnožina reprezentací aminokyselin mezi nimi bude otočena o náhodný úhel kolem osy dané vybranými dvěma pivoty.

Tím by se umožnil pohyb i v rámci smyčky, která vyčnívala ze sbalené struktury u delších řetězců.

Vhodné by bylo udělat pravděpodobnějším ”svihadlový” pohyb pro méně vzdálené pivoty. Tak by se umožnil pohyb řetězce lokálně, oproti globálnímu pohybu v rámci celé struktury, který má menší šanci na přijetí (snadno dojde například k překryvu dvou aminokyselin). Zvýšila by se tím dynamika systému během simulace. Zároveň by se nenarušil princip návrhů stavu, který nemění vzdálenosti mezi aminokyselinami.

Protože bude program používán dál v rámci širšího projektu výzkumného týmu, bylo by vhodné implementovat do něj ještě další vylepšení.

Možnosti vylepšení potenciálových funkcí

Rozlišování mezi jednotlivými typy aminokyselin je v programu zajištěno pomocí tabulky hodnot epsilon využívaných při výpočtu nevazebného, Lennard-Jonesova potenciálu. Pro vylepšení tohoto rozlišování by bylo možné přidat hodnoty poloměrů jednotlivých aminokyselin specificky pro daný typ aminokyseliny.

Jinou možností vylepšení funkce nevazebného potenciálu, může být změna Lennard-Jonesova potenciálu na potenciál získaný pomocí Boltzmannovy inverze z reálných struktur. Podobně už byla tato změna implementována u bending a dihedrálního potenciálu. Tímto by změnilo chování peptidu z hlediska vzdálenosti jednotlivých aminokyselin. Ukázalo se totiž, že (realističtější) funkce vytvořená pomocí Boltzmannovy inverze má dvě minima (lokální minima pro dvě vzdálenosti), nikoli jedno minimum jako Lennard-Jonesův potenciál [19]. Další možná alternativa, Mieův potenciál, již byla zmíněna v kapitole Metody.

Ideální by bylo zkombinovat přístupy zmíněné v předchozích odstavcích, tedy vytvořit potenciálové funkce pomocí Boltzmanovy inverze, které ale budou specifické pro jednotlivé kombinace typů aminokyselin. To by bylo možné udělat i pro úhlové potenciály. Výhledově se tato možnost v projektu celého výzkumného týmu zvažuje. Architektura programu tuto možnost umožňuje, stačilo by pozměnit již existující funkce a přidat načítání dalších vstupních souborů, tentokrát se specifickými potenciály. Seznamy/matice pro nové potenciály je možné dát jako atribut objektu třídy `SimSpace` a použít je při vyhodnocování potenciálové funkce v metodě `pivotMoves` třídy `App`.

Přidání těchto možností by do budoucna mohlo umožnit i vznik sekundárních struktur během simulace.

Zmenšení rozsahu úhlu otočení

Drobností, která by ale mohla pomoci zvýšit podíl přijatých stavů, by bylo zmenšit maximální úhel otočení řetězce během návrhu nového stavu. Testovaných 0.5rad pohnulo s řetězcem často natolik, že u nové konformace došlo například k protnutí reprezentací aminokyselin, což je stav, který téměř není možné přijmout.

Změna verze na Javu 8

Z hlediska implementace simulačního programu by bylo vhodnější program vytvořit v platformě Java verze 8. Zvolená Java 17 sice umožňuje navíc některé funkcionality, které Java 8 nemá. Bohužel ale Java 17 není podporovaná Metacentrem, které je v českém akademickém prostředí využíváno pro gridové výpočty.¹ Výpočty na gridu by umožnily rychlejší simulace většího množství struktur. Proto by bylo vhodné program převést i do Javy 8.

¹Více viz stránky Metacentra, www.metacentrum.cz.

Literatura

- [1] Ken A. Dill, S. Banu Ozkan, M. Scott Shell, and Thomas R. Weikl. The protein folding problem. *Annual Review of Biophysics*, 37(1):289–316, 2008.
- [2] David Beyer, Peter Košovan, and Christian Holm. Simulations explain the swelling behavior of hydrogels with alternating neutral and weakly acidic blocks. *Macromolecules*, 55(23):10751–10760, 2022.
- [3] Christian Anfinsen. Nobel lecture: Studies on the principles that govern the folding of protein chains, 1972. [online; citováno 13. 12. 2022] <https://www.nobelprize.org/uploads/2018/06/anfinsen-lecture.pdf>.
- [4] Giorgia Gambardella, Sara Notari, Dario Cavaterra, Federica Iavarone, Massimo Castagnola, Alessio Bocedi, and Giorgio Ricci. The anfinsen dogma: Intriguing details sixty-five years later. *International Journal of Molecular Sciences*, 23(14), 2022.
- [5] Y Zhang and J Skolnick. The protein structure prediction problem could be solved using the current pdb library. *Proceedings of the National Academy of Sciences of the United States of America*, 102(4):1029–1034, JAN 25 2005.
- [6] Daan Frenkel and Berend Smit. Chapter 4 - molecular dynamics simulations. In Daan Frenkel and Berend Smit, editors, *Understanding Molecular Simulation (Second Edition)*, pages 63–107. Academic Press, San Diego, second edition edition, 2002.
- [7] Florian Weik, Rudolf Weeber, Kai Szuttor, Konrad Breitsprecher, Joost de Graaf, Michael Kuron, Jonas Landsgesell, Henri Menke, David Sean, and Christian Holm. Espresso 4.0 – an extensible software package for simulating soft matter systems. *The European Physical Journal Special Topics*, 227(14):1789–1816, Mar 2019.
- [8] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- [9] Daan Frenkel and Berend Smit. Chapter 3 - monte carlo simulations. In Daan Frenkel and Berend Smit, editors, *Understanding Molecular Simulation (Second Edition)*, pages 23–61. Academic Press, San Diego, second edition edition, 2002.

- [10] Matthew S. Wilson, Guangjie Shi, Thomas Wüst, Ying Wai Li, and David P. Landau. Influence of substrate pattern on the adsorption of hp lattice proteins. *Molecular Simulation*, 44(12):1025–1030, 2018.
- [11] Andrei Neamtu, Francesca Mocci, Aatto Laaksonen, and Fernando L. Barroso da Silva. Towards an optimal monoclonal antibody with higher binding affinity to the receptor-binding domain of sars-cov-2 spike proteins from different variants. *Colloids and Surfaces B-Biointerfaces*, 221, JAN 2023.
- [12] Jakub Nierostek. Počítačové studium skládání proteinů na zjednodušených modelech, bakalářská práce, 2021. [online; citováno 13. 12. 2022] <https://dspace.cuni.cz/bitstream/handle/20.500.11956/128340/130312361.pdf>.
- [13] Lucie Nová and Filip Uhlík. Salt counterion valency controls the ionization and morphology of weak polyelectrolyte miktoarm stars. *Macromolecules*, 55(14):6247–6259, JUL 26 2022.
- [14] Casp 14 - results, 2020. [online, citováno 19. 12. 2022] https://predictioncenter.org/casp14/results.cgi?view=tb_results.
- [15] John Jumper, Richard Evans, Alexander Pritzel, and team. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021-08-26.
- [16] Sebastian Kmiecik, Dominik Gront, Michal Kolinski, Lukasz Wieteska, Aleksandra Elzbieta Dawid, and Andrzej Kolinski. Coarse-grained protein models and their applications. *Chemical Reviews*, 116(14):7898–7936, 2016.
- [17] Guillaume Postic, Nathalie Janel, and Gautier Moroy. Representations of protein structure for exploring the conformational space: A speed–accuracy trade-off. *Computational and Structural Biotechnology Journal*, 19:2618–2625, 2021.
- [18] Siewert J. Marrink, H. Jelger Risselada, Serge Yefimov, D. Peter Tieleman, and Alex H. de Vries. The martini force field: Coarse grained model for biomolecular simulations. *Journal of Physical Chemistry B*, 111(27):7812–7824, JUL 12 2007.
- [19] M. Pavlíková. Nepublikované výsledky bakalářské práce, 2022.
- [20] S Tanaka and HA Scheraga. Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. *Macromolecules*, 9(6):945–950, 1976.
- [21] Stephan Werth, Katrin Stöbener, Martin Horsch, and Hans Hasse. Simultaneous description of bulk and interfacial properties of fluids by the mie potential. *Molecular Physics*, 115(9-12):1017–1030, 2017.
- [22] Jiří Kolafa. *Molekulové modelování a simulace*. VŠCHT, Praha, 2015. [online, citováno 12. 12. 2022] <https://ufch.vscht.cz/files/uze/0014046/y83PKc7MBQA.pdf>.

- [23] Richard Chudoba, Jan Heyda, and Joachim Dzubiella. Temperature-dependent implicit-solvent model of polyethylene glycol in aqueous solution. *Journal of Chemical Theory and Computation*, 13(12):6317–6327, 2017.
- [24] Wikipedie. Otočení — Wikipedie, otevřená encyklopédie, 2022. [online; citováno 13. 12. 2022] <http://cs.wikipedia.org/w/index.php?title=Oto%C4%8Den%C3%AD&oldid=21302050>.
- [25] Lizhong Zhang, He Ma, Wei Qian, and Haiyan Li. Protein structure optimization using improved simulated annealing algorithm on a three-dimensional ab off-lattice model. *Computational Biology and Chemistry*, 85:107237, 2020.
- [26] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.
- [27] BIPM. The International System of Units (SI), 2019. [online, citováno 2. 1. 2023] <https://www.bipm.org/en/measurement-units>.
- [28] Wolfhard Janke. Statistical analysis of simulations: Data correlations and error estimation. *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms, Lecture Notes*, 10:423–445, 2002.
- [29] Miloš Halda. Pivot-move simulation software, 2022. [online, poslední přístup 3. 1. 2023] <https://github.com/ForgoRew/PivotMoveSimulation>.
- [30] Oracle. Jar file specification. [online; citováno 2. 1. 2022] <https://docs.oracle.com/en/java/javase/17/docs/specs/jar/jar.html>.

Seznam obrázků

2.1	Zde jsou zobrazeny schematicky aplikační rozsahy pro různá rozlišení: kvantové, celatomární, zhrubené a ”mesoscale” rozlišení. Na obrázku jsou vidět přibližné rozsahy časových a prostorových jednotek používaných na dané úrovni zjednodušení. Obrázek pochází z práce Kmiecik 2016 [16].	7
2.2	Na obrázku vidíme více typů zjednodušení. (A) je zjednodušení na C_α atomy, (B) C_β , (C) $C_\alpha + C_\beta$, (D) backbone, tedy peptid bez postranních řetězců, (E) backbone + C_β , (F) pouze postranní řetězce, (G) všechny atomy, (H) model ”MARTINI”, který má reprezentace jak backbone tak postranních řetězců, tento model je více popsán v práci Marrink 2007 [18], (I) zjednodušená reprezentace celé struktury, používaná pro zdůraznění sekundárních struktur proteinů. Obrázek převzatý z práce Postic 2021 [17].	8
2.3	Průběh funkce Lennard-Jonesova potenciálu pro zadané parametry.	9
2.4	Funkce bending potenciálu vytvořená pomocí Boltzmannovy inverze Markétou Pavlákovou v rámci její bakalářské práce [19]. Tato potenciálová funkce byla použita k testování simulačního programu.	12
2.5	Na schématu jsou vidět dvě poloviny dané body 1-4. ϕ označuje dihedrální úhel. Obrázek je převzatý z publikace Kolafa 2015 [22].	12
2.6	Graf ukazuje průběh funkce dihedrálního potenciálu použité při testování programu. Funkce byla získána pomocí Boltzmannovy inverze [19] z reálných struktur proteinů. Jsou zde vidět dvě výrazná lokální minima, jedno (globální) v bodě [0.88, -10.36] a druhé v bodě [-2.89, -8.88]. Oblasti kolem těchto minim odpovídají sekundárním strukturám alpha helixům a beta listům.	13
3.1	short	27
3.2	Průběhy potenciálů simulovaných proteinů. Použita pouze data z produktivního části běhu simulace. Legenda u prvního grafu platí pro všechny. Potenciál se mění v důsledku změny teploty.	28
3.3	Barevný kód použitý v ukázkách simulovaných struktur.	28

Seznam tabulek

2.1 Shrnutí volby simulačních jednotek.	22
3.1 Přehled simulovaných proteinů - ID proteinu, délka řetězce (počet aminokyselin v řetězci) a RMSD.	25

Přílohy

Příloha A

Fasta sekvence zkoumaných proteinů

```
>pdb|1dyw|A
MSRSKVFFDITIGGKASGRIVMELYDDVVPKTAGNFRALCTGENGIGKSGKPLHFKGSKFHRIIPNFMI
QGGDFTRGNGTGGESIYGEKFPDENFKEKHTGPGVLSMANAGPNTNGSQFFLCTVKTEWLDGKHVVFGR
VVEGLDVVKAVESNGSQSGKPVKDCMIADCGQLKA
>pdb|1123|A
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSELDKAIGRNCNGVITKDEAEKLFNQ
DVDAAVRAILRNAKLKPVYDSLDAVRRCALINMVFMQMGETGVAGFTNSLRMLQQKRWDEAAVNLAWSRW
YNQTPNRAKRVITTFRTGTWDAYKNL
>pdb|2m1t|A
DTLLGRILPQLVCRLVLCRCSID
>pdb|2msj|A
ANQASVVAQNQLIPINTALTLMRMSEVVTPVGIPAEDIPRLVSMQVSRAVPLGTTLMPDMVKGYAA
>pdb|2nbr|A
GKITFYEDRAFGGRSYETTTDCPNLQPYFSRCNSIRVESGCWMLYERPNYQGQQYLLRRGEYPDYQQWM
GLSDSIRSCCLIPQTVSHRLRYEREDHKGLMMELSEDPSIQDRFHLSEIRSLHVLEGCVWLYPEPNY
RGRQYLLRPQEYRRCQDWGAMDAKAGSLRRVVDLY
>pdb|2ruo|A
GAALQIPFAMQMAYRF
>pdb|2yty|A
GSSGSSGRLLGRNSNSKRLLGYVATLKDNFGFIETANHDKEIFFHYSEFSGDVDSLELGDMVEYSLSKG
KGKVKSAEKVNKTSGPSSG
>pdb|3lj8|A
SFpvqilpnlylgsardsanleslaklgiryilnvtlpnffekngdfhykqipisdhwsqnlsrffp
eaiefidealsqncgvlvhclagvsrvtvaylmqklhlsndaydlvkrkksnispnfnfmqqlld
ferslrle
>pdb|4mq3|A
GSHMGIWQMDCTHFDGKIIIVGIHVESGYIWAQIISQETADCTVKAVLQLLSAHNVTELQTDNGPNFKN
QKMEGVLYMGVKHKFGIPGNPQSQUALVENVNHTLKWWIQQFLPETTSLDNALSLAVHSLNKK
>pdb|5ttt|A
MSWMQNLKNYQHLRDPSEYMSQVYGDPLAYLQETTKFVTEREYYEDFGYGECKNSTESEVQCELITGEF
DPKLLPYDKRЛАWFKEFCYKTSAHGIPMIGEAPLEHHHHHH
>pdb|5wrx|A
VARGWGRKCPPLFG
>pdb|6iws|A
HMLAKEDYYQILGVPRNASQKEIKKAYYQLAKKYHPDTNKDDPKAKEKFSQLAEAYEVLSDEVKRKQYD
AYGS
```