Monte Carlo simulace

1 Úvod

Monte Carlo jsou algoritmy pro simulaci systémů. Jde o stochastické metody[1] používající (pseudo)náhodná čísla. Typicky jsou využívány pro výpočet integrálů, zejména vícerozměrných, kde běžné metody nejsou efektivní. Metoda Monte Carlo má široké využití v matematice, fyzice, chemii, ale i biologii, počítačové grafice, finančnictví a dalších odvětvích. Základní myšlenka této metody je velice jednoduchá, chceme určit střední hodnotu veličiny, která je výsledkem náhodného děje. Vytvoříme počítačový model tohoto děje a po proběhnutí dostatečného množství simulací můžeme data zpracovat klasickými statistickými metodami, třeba určit průměr a směrodatnou odchylku.

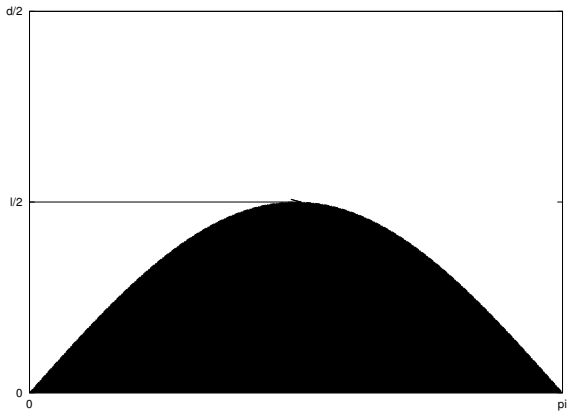
Tato metoda byla formulována ve 40. letech minulého století a je spojena se zkoumáním chování neutronů a projektem Manhattan[2]. Za touto metodou stojí jména jako John von Neumann a Stanislaw Marcin Ulam. Tito dva vědci zkoumali průchod neutronů různým prostředím. I když bylo k dispozici velké množství informací, nedařilo se problém vyřešit ani teoreticky a ani prakticky. Autoři se tedy nechali inspirovat ruletou (proto Monte Carlo). Podle získaných informací věděli, že neutron je pohlcen atomem v jednom případě ze sta. Vybrali si tedy jedno políčko rulety, které symbolizovalo toto pohlcení. Roztočení rulety symbolizovalo pohyb neutronu. V případě, že kulička dopadla na jiné, než označené pole, neutron pokračoval ve své dráze, v případě dopadu na označené políčko byl neutron pohlcen. Tato úvaha je velice jednoduchá a elegantní, její řešení ale nelze provádět roztáčením rulety, zde našly uplatnění počítače.

Pokud půjdeme ještě dále do historie, narazíme na problém formulovaný v roce 1777 francouzským matematikem Georgesem Louisem Lecrerc de Buffonem. Ten formuloval slavnou úlohu známou pod názvem Buffonova jehla. Úloha zní takto:*Na podlaze je velký list papíru, který je rozdělený rovnoběžnými linkami. Vzdálenost mezi všemi linkami je stejná. Na tento papír se libovolným způsobem hází jehla, jejíž délka je rovna vzdálenosti mezi linkami. Jaká je pravděpodobnost, že jehla po dopadu bude ležet tak, že protne některou z linek.*

Pojďme se podívat na řešení úlohy. To jestli jehla protne rovnoběžku závisí na vzdálenosti středu jehly od rovnoběžky (x) a natočení jehly - úhlu, který svírá jehla s rovnoběžkami (φ). Tyto dvě proměnné popisují polohu jehly v rovině a nabývají hodnot $0 \leq \varphi \leq \pi$

[1] Metody, které se zabývají zkoumáním a modelováním náhodných jevů.

[2] Projekt sestrojení atomové bomby, který započal v roce 1942.



Obrázek 1: Grafické znázornění vztahu středu jehly a jejího natočení.

a $0 \leq x \leq \frac{d}{2}$. Jehla délky l protne některou z rovnoběžek právě tehdy, když bude platit $x \leq (\frac{l}{2} \sin \varphi)$. Grafické znázornění souřadnic středu jehly a jejího natočení je vidět na uvedeném obrázku. Vybarvená část vyznačuje kombinaci, kdy jehla protne linku, celý obdélník odpovídá všem možnostem. Pravděpodobnost toho, že jehla protne linku je rovna podílu obsahu vybarvené oblasti vydělené obsahu celého čtyřúhelníku. Obsah vybarvené je integrál $\int_0^\pi \frac{l}{2} \sin \varphi d\varphi = l$ obsah celého čtyřúhelníka je $\pi \frac{d}{2}$ jejich podíl je tedy roven $\frac{2l}{\pi d}$. Vidíme, že v tomto příkladu je pravděpodobnost protnutí rovnoběžky jehlou vyjádřena pomocí čísla π. Pokud provedeme dostatečný počet hodů v tomto experimentu, můžeme vyjádřit číslo π. Matematicky lze tento experiment vyjádřit následovně. Pokud provedeme n hodů a jehla protne rovnoběžky v m případech odpovídá dříve popsaná pravděpodobnost $\frac{2l}{\pi d} = \frac{m}{n}$. Úpravou výrazu vyjádříme výpočet čísla π následovně: $\pi = \frac{2ln}{md}$, l je délka jehly, d je vzdálenost mezi rovnoběžkami n je celkový počet hodů a m je počet hodů, kdy jehla protne rovnoběžku. Uvedený pokus výpočtu čísla π popsal v roce 1873 A. Hall. Jeho popis vyzkoušel prakticky v roce 1850 pan Volf a při 5000 hodech určil číslo π na hodnotu 3,1596. Takové určování je ale velmi zdlouhavé, proto jsou užívány počítače.

Všechny následující úvahy se opírají o matematické zákony: Zákon velkých čísel a centrální limitní větu.

Zákon velkých čísel je několik podobných matematických vět z oblasti teorie pravděpodobnosti tvrdících, že aritmetický průměr n náhodných veličin se stejnou střední hodnotou se s rostoucím n za určtých předpokladů blíží k této střední hodnotě.

Centrální limitní věta v teorii pravděpodobnosti označuje tvrzení, podle něhož se (za určitých podmínek diskutovaných níže) rozdělení výběrového průměru po vhodné normalizaci blíží k normálnímu rozdělení [3] . O náhodné veličině s uvedeným chováním říkáme,

[3]Normální rozdělení neboli Gaussovo rozdělení (podle Carla Friedricha Gausse) je jedno z nejdůležitějších rozdělení pravděpodobnosti spojité náhodné veličiny. (Slovo „normální“ zde není použito v nejběžnějším smyslu „obyčejné, běžné“, ale znamená „řídící se zákonem, předpisem nebo modelem“.) Jeho důležitost ukazuje centrální limitní věta (CLV), jež zhruba řečeno tvrdí, že součet či aritmetický průměr velkého počtu libovolných vzájemně nezávislých a nepříliš „divokých“ náhodných veličin se vždy podobá

že má asymptoticky normální rozdělení.

2 Obecný postup metody Monte Carlo

Nejčastější postup metody Monte Carlo je modelování takové náhodné veličiny X, že její střední hodnota $E(X)$ je rovna hledané hodnotě a. Tedy, abychom mohli přibližně vypočítat veličinu a, musíme najít takovou náhodnou veličinu X, že platí $E(X) = a$.

Jestliže tedy n nezávislých realizací $X_1; X_2; \ldots X_n$ náhodné veličiny X, můžeme odhadnout hodnotu a pomocí aritmetického průměru

$$a \doteq \frac{1}{n}(X_1 + X_2 + \ldots + X_n)$$

Obecně platí, že existuje nekonečně mnoho náhodných veličin X takových, že $E(X) = a$. Proto teorie metody Monte Carlo musí odpovědět na následující dvě otázky:

1. Jak vybrat náhodnou veličinu X, lépe řečeno, jak vybrat nejvhodnější náhodnou veličinu X pro výpočet hledané veličiny?

2. Jak najít hodnoty $x_1, x_2, \ldots x_n$ libovolné náhodné veličiny X?

Ke druhé otázce lze říci, že v převážné většině MC simulací se postupuje tak, že se nejdříve generují hodnoty náhodné veličiny Y rovnoměrně rozdělené na intervalu $(0, 1)$ a ty se pak transformují pomocí vhodné funkce f na hledané hodnoty x.

$$x_i = f(y_i, y_{i-1}, \ldots)$$

Výpočet hodnot x_i nemusí být zadán přímo funkcí, ale může se provádět pomocí vhodného algoritmu. Druhou otázku můžeme tedy zredukovat na otázku generování náhodných čísel rovnoměrně rozdělených na intervalu $(0, 1)$ a nalezení vhodné transformace. Obecný postup Monte Carlo metody lze rozdělit do následujících kroků:

1. Generování náhodných čísel y_i s rovnoměrným rozdělením na intervalu $(0, 1)$.

2. Transformace náhodných čísel y_i na náhodná čísla z_i se složitějším rozdělením.

3. Pomocí náhodných čísel z_i se buď přímo počítají odhady charakteristik náhodné veličiny X nebo se počítají pomocí vhodného algoritmu hodnoty x_i a odhady charakteristik náhodné veličiny.

4. Získané výsledky se zpracují statistickými metodami.

normálně rozdělené náhodné veličině. Normální rozdělení proto za určitých podmínek dobře aproximuje řadu jiných pravděpodobnostních rozdělení (spojitých i diskrétních), i když v praxi málokteré rozdělení je přesně normální.

3 Generování náhodných čísel s rovnoměrným rozložením

Monte Carlo simulace předpokládá modelování náhodného procesu pomocí operací s náhodnými čísly. Podmínkou úspěchu je možnost náhodný proces mnohonásobně opakovat a k tomu je třeba mít k dispozici dostatečný počet náhodných čísel s nejrůznějšími distribučními funkcemi. Pro každou distribuční funkci není zapotřebí tvořit speciální generátor náhodných čísel, který odpovídá dané distribuční funkci. V praxi si vystačíme s generátorem náhodných čísel s rovnoměrným rozdělením na intervalu $(0, 1)$. Náhodné veličiny s jiným typem rozdělení lze potom získat vhodnou transformací.

Proto jsou nejdůležitější součástí metody Monte Carlo generátory náhodných čísel[4]. Generátory náhodných čísel s definovaným stochastickým rozdělením jsou základem simulačních programů Monte Carlo. Generátory náhodných čísel fungují tak, že nejprve vygenerují posloupnost náhodných čísel s rovnoměrným rozdělením (primární generátor) a poté je z nich transformací vytvořena posloupnost s požadovaným rozdělením. Primární generátory jsou fyzikální nebo pseudonáhodné.

3.1 Fyzikální generátory náhodných čísel

Jako fyzikální generátor náhodných čísel se dá chápat vše, co má charakter náhodnosti, jako např. házení kostkou nebo mincí. Častěji se ale používají generátory založené na kombinaci radioaktivního zářiče a detektoru (vyzařuje částice v náhodném množství). Fyzikální generátory se ale málo používají, jelikož mohou být v závislosti na vnějších vlivech nestabilní, posloupnost čísel se nedá zopakovat a chování se může lišit v důsledku výrobních tolerancí.

3.2 Pseudonáhodné generátory náhodných čísel

Generátor pseudonáhodných čísel je efektivní deterministický program, který generuje posloupnost čísel, statistickými testy pokud možno nerozlišitelnou od náhodné. Byť existují zdroje skutečně náhodných jevů (kvantové generátory, šum), pseudonáhodné generátory (a postupy jakými se vytvářejí) jsou klíčovým prostředkem moderní kryptografie. Na nich se zakládají pravděpodobnostní kryptosystémy s veřejným klíčem, digitální podpisová schémata, bit-commitment protokoly a interaktivní zero-knowledge důkazové systémy.

Vstupními daty pro pseudonáhodné generátory jsou skutečně (téměř) náhodné (pokud probíhá útok postranním kanálem navíc záměrně ovlivňované), leč krátké, posloupnosti zvané random seed, které jednoznačně určují další běh programu (generátoru). V důsledku deterministočnosti těchto programů jsou na počítači s ohraničenou pamětí nevyhnutelně periodické, tedy po určité době (periodě) se generovaná posloupnost začne opakovat. Ta však může být velmi dlouhá, tudíž nedetekovatelná. Standardizované generátory ovšem mohou

[4]Generátor náhodných čísel je výpočetní nebo fyzické zařízení, které generuje řadu náhodných čísel, tj. čísel která postrádají jakýkoliv vzor (sekvenci, předvídatelnost).

obsahovat posloupnosti vytvářející u šifrovacích algoritmů „zadní vrátka“ tj. univerzální klíč.

Je otázkou, je-li možné současnými výpočetními prostředky rozlišit náhodnou posloupnost od posloupnosti pseudonáhodné, pokud nedisponujeme znalostí „random seed“ a použitého algoritmu generátoru.

Generátor pseudonáhodných čísel je nastaven do jakéhokoliv výchozího stavu použitím random seed (náhodné číslo – počáteční stav). Poté vždy vyprodukuje stejnou sekvenci čísel, pokud je inicializován se stejným počátečním stavem. Perioda generátoru pseudonáhodných čísel je definována jako maximum nad všemi počátečními stavy délky neopakující se sekvence. Perioda je omezena velikostí stavu měřeného v bitech. Nicméně od té doby, co se délka periody potenciálně zdvojnásobuje s každým stavovým bitem, je snadné vytvořit generátor pseudonáhodných čísel s periodou dostatečně dlouhou pro praktické aplikace.

Pokud vnitřní stav generátoru pseudonáhodných čísel obsahuje n bitů, pak jeho perioda nemůže být delší než $2n$ výsledných stavů, ale může být mnohem kratší. Pro některé generátory je možné délku periody vypočítat bez procházení skrze celou periodu. Posuvný registr s lineární zpětnou vazbou je obvykle volen s periodou $2n - 1$. Lineární kongruentní generátor mívá periodu, která může být vypočítána faktorizací. Přestože generátor pseudonáhodných čísel bude opakovat výsledky poté, co dosáhne konce periody, opakovaný výsledek nezaručuje dosažení konce periody, poněvadž jeho vnitřní stav může být větší než výsledný. To je zřejmé zejména u generátoru pseudonáhodných čísel s 1 bitovým výstupem.

Většina algoritmů pseudonáhodných generátorů produkuje sekvenci s rovnoměrným rozdělením.

Problémy s deterministickými generátory

V praxi výstup z mnoha běžných generátorů pseudonáhodných čísel vykazuje anomálii, která způsobuje jejich selhání při statistické detekci vzoru. Například:

- periody pro některé výchozí stavy jsou kratší než očekáváme (takové stavy mohou být v tomto kontextu nazvány „slabými“)
- chybějící rovnoměrnost rozdělení pro velké množství generovaných čísel
- korelace po sobě následujících hodnot
- slabá dimensionální distribuce výsledné sekvence
- rozdíl mezi tím, jak jsou určité hodnoty distribuovány od těch s náhodnou distribucí

Chyby vyskytující se ve vadných generátorech pseudonáhodných čísel se objevují od těch nejnepatrnějších (a neznámých) až ke zřejmým. Jako příklad může být uveden RANDU, což je algoritmus náhodných čísel, používaný po celá desetiletí na mainframech[5] . Tento algoritmus byl vskutku nedostatečný, ale jeho neadekvátnost zůstala bez povšimnutí po celá

[5]Mainframe (sálový počítač, střediskový počítač) je počítač používaný převážně velkými firmami pro kritické aplikace, často zahrnující zpracovávání velkých objemů dat. Mezi typické úlohy zpracovávané mainframy patří sčítání lidu, rozsáhlé statistické úlohy, ERP nebo finanční transakce. Většinou se jedná o sálové počítače, které byly konstruovány do enormních rozměrů v řádu místností.

léta. V mnoha oborech, bylo velké množství výzkumných prací té doby, která se spoléhala na náhodný výběr nebo na metodu Monte Carlo, jinak řečeno jsou méně spolehlivé než by mohly být v případě výsledků.

3.2.1 Nejpoužívanější generátory pseudonáhodných čísel

Mersene Twister generátor Mersenne Twister je generátor pseudonáhodných čísel vyvitutý Makoto Matsumotem a Takuji Nishimurou. Tento generátor nevyužívá aritmetických operací ve smyslu sčítání, odečítání, násobení či dělení. Využívá pouze bitového posunu a funkcí and, or a xor. Perioda tohoto generátoru je $2^{19937} - 1$. Tento generátor je díky použitým operacím velice rychlý. Úspěšně prošel řadou testů statistické náhodnosti a je hojně využíván.

Lineární kongruentní generátor Lineární kongruentní generátor patří mezi nejjednodušší generátory. Je založen na prin cipu algoritmu 11 se stavy $S_i = X_i \in 0, \ldots, m-1$, pro nějaké kladné m , které nazýváme modulem. Přechody mezi stavy jsou definovány jako

$$X_i = (a * x_{i-1} + c) mod\, m,$$

pro

$$i = 1, 2, \ldots,$$

kde činitel a a přírůstek c jsou celá čísla. Pokud je $c = 0$, je občas generátor nazýván jako multiplikativní. Většina existujících lineárních kongruentních generátorů je v této formě. Přírůstek c totiž nemá příliš velký vliv na kvalitu generátoru. Výstupní funkce generátoru vypadá takto:

$$U_i = \frac{X_i}{m}$$

Například lineární kongruentní generátor jazyka C++ má periodou 2147483647.

Subtract with carry Subtract with carry je zpožděný Fibonacciho generátor, který je definován rekurentně:

$$x_i = x_{i-S} - x_{i-R} - cy_{i-1} mod\, M,$$

kde

$$cy_i = \begin{cases} 1, & \text{pro } x_{i-S} - x_{i-R} - cy_{i-1} < 0 \\ 0, & \text{jinak} \end{cases}$$

Konstanty S a R se nazývají krátká a dlouhá prodleva, výrazy x_{i-S} a x_{i-R} korespondují s S-tým a R-tým členem sekvence. S a R splňují podmínku $0 < S < R$. Modulo M nabývá hodnoty $M = 2^W$ kde W je bitová délka klíčového slova. Teorie z hlediska matematiky je u tohoto generátoru zatím nejasná. Vlastnosti jsou odvozeny pouze ze statistických výsledků, ve kterých vychází lépe než Lineární kongruentní generátor. Tento generátor je jeden ze tří standardních generátorů knihovny jazyka C++.

3.3 Rozšíření klasické Monte Carlo metody

V praxi můžeme narazit na problém, kdy náhodné generování stavů v mnoha případech povede k nesmyslným stavům a to v převládajícím množství. Potom bychom i při velkém počtu provedených experimentů nedošli ke správným výsledkům. Příkladem může být rozmístění n tuhých koulí (atomů) do zadaného objemu. Pokud dojde k překryvu alespoň dvou koulí, je takový stav nemožný. Možné stavy jsou pouze v případě, že se žádné dvě koule nebudou překrývat. Proto je třeba klasickou MC „vylepšit“. V takovém případě vyjdeme z vhodného stavu a změnou konfigurace tohoto stavu dojdeme k další konfiguraci. Vytváříme tzv. Markovovy řetězce, nebo aplikujeme Metropolisův algoritmus, který z Markovových řetězců vychází.

3.3.1 Markovův řetězec

Markovův řetězec popisuje obvykle diskrétní náhodný (stochastický či pravděpodobnostní) proces, pro který platí, že pravděpodobnosti přechodu do následujícího stavu závisejí pouze na současném stavu, ne na předchozích stavech. Tato tzv. markovovská vlastnost dovoluje proces znázornit stavovým diagramem, kde z každého stavu (uzlu grafu) vycházejí hrany možných přechodů do dalšího stavu s připsanou pravděpodobností.

Jinými slovy Markovův řetězec je posloupnost náhodných veličin $X^{(k)}$, kde $k = 1, 2, \ldots \infty$ taková, že událost, kterou pozorujeme v kroku $k+1$, závisí na tom, jaká událost byla pozorovaná v kroku k. Tedy, jestliže se v čase k vyskytne událost A_i s pravděpodobností $\pi_i^{(k)}$, (náhodná veličina $X^{(k)}$ nabývá hodnoty A_i s pravděpodobností $\pi_i^{(k)}$), pak v čase $k+1$ se událost A_j vyskytne s pravděpodobností, pro kterou platí

$$\pi_j^{(k+1)} = \sum_{i=1}^{M} \pi_i^{(k)} W_{i \to j}$$

Zapsáno vektorově: $\pi = (\pi_1, \pi_2, \ldots, \pi_M$, kde M je počet stavů.

$$\pi^{(k+1)} = \pi_k . W$$

kde veličina W se nazývá **matice přechodu**. Prvky matice přechodu mají fyzikální význam pravděpodobnosti přechodu ze stavu A_i do stavu A_j. Součet prvků v každém řádku matice musí být roven 1.

Markovovým řetězcem může být například posloupnost hodů hrací kostkou. V tomto případě můžeme připustit, že při jednom hodu je číslo, které padne, ovlivněno předchozí polohou kostky (nedostatečné protřepání kostky v dlani), není však nijak ovlivněno polohou před dvěma hody.

Příklad

Pravděpodobnost výskytu jevu v bodě 3 můžeme vyjádřit následovně:

$$\pi^{(2)} = \pi^{(1)} . W$$

a

$$\pi^{(3)} = \pi^{(2)}.W = (\pi^{(1)}.W).W = \pi^{(1)}.W^2$$

Příklad Máme systém ve dvou stavech A, B. Pravděpodobnost, že systém A přejde do systému B je 90%, pravděpodobnost, že systém zůstane v tomto stavu je tedy 10%. Pravděpodobnost, že systém přejde z konfigurace B do konfigurace A je 30%, pravděpodobnost, že zůstane va stejné konfiguraci je 70%. Stav pravděpodobnosti, že se nacházíme v určité konfiguraci můžeme popsat vektorem $\pi^{(k)} = (\pi_A^{(k)}, \pi_B^{(k)})$. Matici přechodu W tvoří vektory (0.9, 0.1) a (0.3, 0.7). Pokud se nacházíme v konfiguraci A, je $\pi^{(1)} = (1, 0)$ v dalším kroku je pravděpodobnost výskytu ve stavu A popsaná vektorem $\pi^{(2)} = (0.9, 0.1)$. Nachází-li se systém v konfiguraci B, $\pi^{(1)} = (0, 1)$, je pravděpodobnost v následujícím kroku $\pi^{(2)} = (0.3, 0.7)$. V následujícím kroku bude pravděpodobnost pro první případ (vycházíme z konfigurace A) $\pi^{(3)} = (0.84, 0.16)$. Vycházíme-li z konfigurace B, je pravděpodobnost výskytu dána vektorem $\pi^{(3)} = (0.48, 0.52)$. Provedeme-li velké množství generování transformací, blíží se k limitně k nekonečnu, dojdeme k limitnímu rozložení $\pi^{(n)} = \pi = (0.75, 0.25)$.

3.3.2 Metropolisův algoritmus

V reálném případu ale řešíme problém s více než dvěma stavy. Proto musíme použít ještě sofistikovanější metody. Metropolisův algoritmus vychází z metod Markovových řetězců a hodí se právě pro analýzu systémů s více než dvěma možnými stavy. Celý algoritmus můžeme popsat pomocí 4 následujících kroků:

1. Vybereme libovolnou částici i, (výběr můžeme provést náhodně).

2. Zkusíme náhodně změnit její polohu,

$$x_i^{new} = x_i + u_{(-d,d)}$$

$$y_i^{new} = y_i + u_{(-d,d)}$$

$$z_i^{new} = z_i + u_{(-d,d)}$$

   $u_{(-d,d)}$ je náhodné číslo rovnoměrně rozdělené v intervalu $(-d,\ d)$. To znamená, že pravděpodobnost opačného pohybu je stejná.

3. Vypočítáme změnu potenciální energie systému $\Delta U = U^{new} - U$.

4. Je-li $\Delta U \leq 0$, změnu přijmeme a pokračujeme s novou konfigurací. Je-li $\Delta U > 0$, změnu přijmeme s pravděpodobností $e^{-\beta \Delta U}$. To znamená, že pokračujeme se stejnou konfigurací při pravděpodobnosti $1 - e^{-\beta \Delta U}$.

Uvedené kroky opakujeme mnohokrát.

3.3.3 Simulované žíhání

Tato metoda je úpravou Metropolisova algoritmu. Je určena k hledání globálního minima funkce. Stejně jako u Metropolisova algoritmu pro výpočet energie figuruje teplota systému. Ta je v případě Metropolise konstatní po celou dobu simulace. V případě simulovaného žíhání je na počátku teplota nastavena na vysokou hodnotu, což umožní přijetí téměř každé konfigurace systému, neboť je velmi málo konfigurací zamítnuto. Postupně dochází ke snižování teploty a tím jsou vybírány konfigurace s nižší energií. Na konci simulace je už přijata pouze jedna energeticky nejvýhodnější simulace.

3.4 Aplikace Monte Carlo v počítačové chemii

Monte Carlo metoda je snadno implementovatelná na atomární systém, protože zde bereme v úvahu translační stupně volnosti. (Atomy bereme jako rigidní koule.) Algoritmus je snadno implementovatelný a výsledky získáme pro relativně malý počet simulačních kroků (desítky tisíc). Praktické problémy vyvstanou při aplikaci MC na molekulární systémy, zvláště pak na flexibilní molekuly s mnoha stupni volnosti. Zde je třeba zahrnout vnitřní stupně volnosti molekul, které musíme měnit. To vede v mnoha případech ke konfiguracím s velmi vysokou energií, které bývají zamítnuty. Musíme tedy provádět mnohonásobně větší počet MC simulací.

Rigidní molekuly V případě rigidních nesférických molekul musíme brát v úvahu jak jejich orientaci, tak jejich vzájemnou pozici v simulovaném prostoru. V praxi se to provádí tak, že molekulu v prostoru posuneme a také s ní otočíme v průběhu jednoho MC simulačního kroku. Posun je popsaný pomocí změny polohy těžiště molekuly. Pro popis otočení molekuly máme několik možností[6]. Například můžeme zvolit kombinaci rotací kolem souřadných os. Zahrnutí rotace společně s translací vede k většímu počtu zamítnutých konfigurací a je tedy třeba provádět mnohem větší počet MC simulací, abychom splnili požadovaná kritéria modelu.

MC simulace flexibilních molekul MC simulace flexibilních molekul jsou, při použití klasické MC, možné pouze pro malé molekuly. Pro simulaci velkých flexibilních systémů musíme využít určitá „vylepšení metody“, jako například zafixování určitých rigidních částí molekuly. Nejjednodušší cesta změn poloh atomů zde není použitelná, proto je vhodné použít pro popis molekuly tzv. vnitřních souřadnic [7]. Zde opět si znovu připomeneme, že i malá změna torzního úhlu uprostřed molekuly může vést k rozsáhlým změnám souřadnic atomů vzdálených od atomů rotující torze. Tato změna často vede k překryvu atomů v molekule a prudkému vzrůstu energie systému, což vede k zamítnutí změny. Zde platí přímá úměrnost mezi počtem stupňů volnosti systému a počtem Monte Carlo kroků.

[6] Připomeňme si matice posunu, zrcadlení, rotace, o kterých jsme se bavili v předchozích přednáškách.
[7] Bližší informace naleznete v přednášce o použití matic.

Modely použitelné pro MC simulace polymerů Polymery jsou makromolekuly, které se skládají z opakujících se jednotek (monomerů), chemicky navzájem vázaných do molekuly. Některé polymery jsou tvořeny řetězci jednoho monomeru (polyethylen, polystyren). Jiné polymery jsou tvořeny směsí různých monomerů, příkladem jsou například bílkoviny, které jsou tvořeny dvaceti základními aminokyselinami. Spojení různých jednotek vede k rozdílným vlastnostem v dané části polymeru. Každá jednotka monomeru má své specifické konformační vlastnosti, což ovlivňuje celkové chování molekuly. Zmapování chování polymeru za různých podmínek je důležité pro praktické využití. Protože se jedná o simulace velkých systémů s mnoha stupni volnosti, nemůžeme pro generování použít klasické MC metody. Pro výpočet energií stavů jsou použity empirické modely založené na zjednodušených aproximacích, narůstající výpočetní výkon počítačů nám sice umožňuje studovat stále větší systémy, ale stále je třeba využívat další „smartness“.

Jednou z možností je použití tzv. **modelu mřížkové stavby polymeru**. Kde do jednotlivých bodů na mřížce (2D, 3D)umísťujeme monomerní jednotky s popsanými vlastnostmi a hodnotíme, zda konfiguraci jednotek přijmeme nebo zamítneme. Jednou ze základních podmínek přijetí je, že generovaný řetězec se nesmí na mřížce sám protnout. Tento způsob je velice rychlý a efektivní a dokáže vygenerovat a ohodnotit velké množství stavů pro velké systémy.

Další model **kontinuální model polymeru** staví polymer postupně z jednotek (můžeme si jej představit jako navlékání korálků na šňůru). Přidané jednotky (korálky) interagují se svými navázanými sousedy, ale také se svým okolím (nevazebné interakce). Zde ale opět narážíme na značnou různorodost stavů díky počtu volně rotovatelných vazeb, a proto je nutná aplikace dalších algoritmů, které umožní řešení problému.

Narůstající výpočetní výkon současných počítačů nám umožňuje provádět simulace velkých systémů za přítomnosti molekul solventu i za přítomnosti iontů. Při těchto simulacích se běžně používá kombinace molekulové dynamiky s Monte Carlo metodou. Ta je použita pro generování a optimalizaci pozic molekul solventu a iontů. Při této optimalizaci je studovaná molekula polymeru držena v tzv. zamrzlé pozici. Volba metody, použité aproximace, použití různých algoritmů vždy záleží na tom, jakou vlastnost molekul chceme studovat.