VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

EFEKTIVNÍ ANALÝZA STOCHASTICKÝCH BIOCHEMICKÝCH SYSTÉMŮ

EFFICIENT ANALYSIS OF STOCHASTIC BIOCHEMICAL SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUCIA TUŠIMOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MILAN ČEŠKA, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21187/2017/xtusim00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Tušimová Lucia**
Obor: Informační technologie
Téma: **Efektivní analýza stochastických biochemických systémů**
Efficient Analysis of Stochastic Biochemical Systems
Kategorie: Formální verifikace

Pokyny:

1. Seznamte se s problematikou modelování a analýzy biochemických systémů pomocí stochastických modelů a kvantitativních formální metod.
2. Nastudujte existující aproximační techniky (zejména fast adaptive uniformization) dovolující analyzovat složité biochemické modely.
3. Implementujte techniku fast adaptive uniformization v rámci nástroje STORM či PRISM.
4. Experimentálně ověřte praktickou užitečnost implementované techniky na vhodné sadě modelů. Zaměřte se zejména na efektivitu implementace v kontextu velikosti verifikovaných modelů.

Literatura:

- M. Mateescu, V. Wolf, F. Didier, and T. A. Henzinger. Fast Adaptive Uniformization of the Chemical Master Equation. *IET Systems Biology*, 4(6):441-452, 2010.
- M. Kwiatkowska, G. Norman and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *CAV'11*, LNCS, pages 585-591, Springer, 2011.
- Ch. Dehnert, S. Junges, J.P. Katoen, and M. Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *CAV'17*, LNCS, pages 592-600, Springer, 2017.

Pro udělení zápočtu za první semestr je požadováno:

- První dva body ze zadání a prototypová implementace aproximačních technik pro stochastické biochemické modely.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Češka Milan, RNDr., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Cieľom tejto práce je zefektívniť výpočet biochemických reakčných sietí. Na modelovanie biochemických systémov pomocou numerických metód sa používajú Markovove reťazce v spojitom čase. Problémom pri biochemických reakciách je vznik nezvládnuteľného množstva stavov. Metóda rýchlej adaptívnej uniformizácie rieši tento problém za cenu malej zaokrúhľovacej chyby. Táto metóda bola implementovaná v nástroji STORM, ktorý slúži na analýzu systémov obsahujúcich pravdepodobnostné javy. Následne bola efektivita tejto metódy overená na sade experimentov.

Abstract

The aim of the thesis is to make computation of the biochemical reaction networks more e□cient. Modeling of biochemical systems using numerical methods uses continuous-time Markov chains. The problem is that in biochemical reactions arises an unmanageable amount of states. The fast adaptive uniformization method solves this problem at the cost of a small rounding mistake. This method was implemented in the STORM, which is tool for the analysis of systems involving random or probabilistic phenomena. Consequently, the e□ectiveness of this method was verified on a set of experiments.

Kľúčové slová

Biochemické modely, Markovove reťazce v spojitom čase, Rýchla adaptívna uniformizácia, STORM

Keywords

Biochemical models, Continuous-time Markov chain, Fast adaptive uniformization, STORM

Citácia

TUŠIMOVÁ, Lucia. *Efektivní analýza stochastických biochemických systémů*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Milan Češka, Ph.D.

Efektivní analýza stochastických biochemických systémů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána RNDr. Milana Češku Ph.D. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

.....
Lucia Tušimová
17. mája 2018

Poďakovanie

V úvode by som rada poďakovala vedúcemu mojej práce RNDr. Milan Češka Ph.D. za odborné vedenie, cenné rady, venovaný čas a trpezlivosť pri konzultáciách práce.

Obsah

Kapitola 1

Úvod

Fyzici, chemici či biológovia pri svojej vedeckej činnosti od nepamäti využívajú experimenty, ktoré sú často veľmi náročné z finančného, časového a mnohokrát aj z bezpečnostného hľadiska. S príchodom počítačov a výpočtovej techniky sa však otvorajú nové možnosti.

V súčasnosti sa už na množstvo experimentov používajú počítačové simulácie. Reálny systém nahradíme počítačovým modelom a následne na ňom vykonáme experimenty. Táto metóda sa používa v mnohých odvetviach cez meteorológiu, strojárstvo, až po biochémiu. Pomocou simulácií vieme získať nové poznatky o reálnom systéme. Výhodou je napríklad možnosť urýchliť čas experimentu. Nemusíme čakať niekoľko rokov kým vyrastie les v prírode, môžeme to jednoducho simulovať. I keď je toto odvetvie široko využívané, stále má svoje úskalia, medzi ne patrí aj náročnosť na výpočtový výkon počítača.

V mojej práci sa budem zaoberať spomínanými biochemickými systémami. Táto veda skúma chemické reakcie, zlúčeniny, látky a deje v živých organizmoch. Konštruovanie biochemických počítačových modelov sa používa už od roku 1969 [8] a stále je aktuálnou témou [2]. Tieto procesy sú náročné na spracovanie, pretože prebiehajú v reálnom čase a sú náhodné. Nevieme presne určiť, kedy a ktorá reakcia nastane.

Predstavíme si Markovove reťazce v spojitom čase[14], pomocou ktorých môžeme vhodne popísať biochemickú reakčnú sieť, ktorá sa skladá z chemických reakcií a ich rýchlostí. Napríklad to môže, byť génová expresia, rozhodnutia o osude buniek alebo cirkadiánne kmity. Markovov reťazec reprezentuje chemické populácie ako náhodné premenné. Jeho vývoj je daný systémom lineárnych obyčajných diferenciálnych rovníc, známych ako základná chemická rovnica.[17] Avšak veľkosť Markovovho reťazca v spojitom čase, ktorú predstavuje biochemická reakčná sieť, zvyčajne presahuje to, čo je uskutočniteľné. Toto je ďalší problém biochemických systémov, takzvaná stavová explózia. V tejto práci si predstavíme metódu rýchlej adaptívnej uniformizácie [16], ktorá za cenu malej aproximačnej chyby zvládne tento výpočet. Následne experimentálne vyhodnotíme jej praktickú užitočnosť.

Ako nástroj do ktorého budem implementovať rýchlu adaptívnu uniformizáciu, som si zvolila Storm [6]. Patrí medzi najvýkonnejšie nástroje pre pravdepodobnostné overovanie modelov, avšak v súčastnosti neobsahuje túto metódu.

Prvá väčšia časť práce je prevažne teoretická predstavíme si biochemické systémy, Markovove modely, štandardnú a adaptívnu uniformizáciu. Následne si vysvetlíme s využitím predchádzajúcej teórie rýchlu adaptívnu uniformizáciu. V druhej polke práce využijem nadobudnuté vedomosti. Stručne opíšeme nástroj Storm a implementáciu. Potom budem postupne na rôznych experimentoch ukazovať vlastnosti rýchlej adaptívnej uniformizácie. Predstavím jej výhody a nevýhody v porovnaní s pôvodným výpočtom nástroju Storm ale aj voči štandardnej uniformizácii.

Kapitola 2

Modelovanie biochemických systémov

Počítačovým modelovaním sa snažíme napodobiť prvky, deje reálneho sveta. Modelovanie systémov je založené na transformovaní reálneho systému do matematického modelu. Ten môže byť napríklad Petriho sieť, teória automatov, diferenciálne procesy alebo Markovove procesy. Modelovať je možné čokoľvek, ale musíme charakterizovať kľúčové správanie systému.

Po vytvorení modelu je dôležité ho verifikovať a validovať či sa vlastnosti modelu zhodujú so systémom. S modelovaním je úzko spätá simulácia, inak povedané experimentovanie s modelom. Pomocou experimentov s modelom získavame nové informácie o systéme, ktoré je potrebné analyzovať. Výhodou simulácií je možnosť ovplyvňovať celý systém skrz zmenu jednotlivých parametrov v rámci modelu.

Modelovanie a simulácia je aplikovaná v mnohých odvetviach cez ekonomiku, psychológiu či strojárstvo. Využívajú sa ako náhrada za drahé, nebezpečné a časovo náročné experimenty. Príkladom môže byť jednoduchý štatistický softvérový program, používaný pri obchodnej analýze, ale i náročné simulácie počasia na ktoré sú zvyčajne využívané superpočítače. Simulácie riek môžu pomôcť pri rozhodovaní o potrebnosti vodných nádrží. Ďalšie možnosti pre modelovanie môžu byť jadrové výbuchy alebo šírenie infekčných chorôb.

Modelovanie a simulácia svoje uplatnenie nachádza aj v biochémii. Biochemické systémy sú vo svojej podstate veľmi zložité, keďže sa v nich vyskytujú súčasne mnohé reakcie alebo procesy. Chovanie takýchto systémov sa často musí skúmať nepriamymi metódami, ktoré vyžadujú manipuláciu s veľkým množstvom údajov, ktoré sa často nedajú spracovať pomocou manuálnych výpočtov. Meniace sa dáta môžu viesť k veľkému počtu nelineárnych diferenciálnych rovníc. Prirodzenou odpoveďou na túto situáciu je využívanie počítačov a modelovania.

Výpočtový systém v biológii môže na základe matematických prístupov vyvinutých v kontexte informatiky a techniky prispieť k vytvoreniu silných nástrojov na simuláciu, analýzu a argumentáciu pre biológov. Tieto nástroje môžu byť použité pri navrhovaní nových experimentov a v konečnom dôsledku pre pochopenie funkčných vlastností buniek a organizmov.

2.1 Biochemický systém

Všetky biologické systémy, od jednotlivých biochemických ciest po mnohobunkové organizmy, možno považovať za veľmi zložité systémy. Tieto systémy môžeme brať ako reakčné, pretože neustále vytvárajú interakcie s ich prostredím. Systémová biológia skúma zložité interakcie v biologických systémoch s cieľom lepšie pochopiť procesy, ktoré sa vyskytujú v takomto systéme, ako aj pochopiť nové vlastnosti takého systému ako celku [5].

Na popis biologických systémov sú široko používané biochemické reakčné siete. Tie sa skladajú zo zoznamu chemických reakcií a súvisiacimi reakciami.

$$2H_2O \xrightarrow{1} 2H_2 + O_2$$

Rýchlosť 1 z chemickej reakcie znázorňuje tendenciu uskutočniť reakciu. Tieto siete sa využívajú pri numerických analýzach rozdelenia pravdepodobnosti medzi všetky možné stavy, ktoré sú témou tejto práce. V poslednej dekáde zaznamenali stochastické modely s diskrétnym stavovým priestorom zväčšujúci sa záujem , pretože poskytujú primeraný popis systémov s molekulárnym šumom. Tento šum vyplýva z náhodnostných udalostí v bunke, ktoré významne ovplyvňujú základne biologické procesy [17].

2.2 Základná chemická rovnica

Základnú chemickú rovnicu (angl. chemical master equation) budeme ďalej v texte označovať ako CME. Je to súbor lineárnych, autonómnych, obyčajných diferenciálnych rovníc. Jedna diferenciálna rovnica je definovaná pre každý možný stav systému. Riešenie $x$tej rovnice nám určuje pravdepodobnosť $p(x,t)$, že systém sa nachádza v danom stave x v čase t. CME opisuje časový vývoj pravdepodobnosti $p(x,t)$. Podstatné je, že počet diferenciálnych rovníc nie je daný počtom druhov, ale počtom možných stavov systému.

Systémy ktorými sa budeme zaoberať budú obsahovať rôzne typy molekúl alebo chemických druhov. Ich počet bude vyjadrený pomocou premennej N. Tieto molekuly sa môžu zúčastniť na jednotlivých alebo viacerých typoch chemických reakcií R. Stav systému sa môže meniť pomocou ktorejkoľvek z reakcii M teda R : μ 2 f1, 2, ..., Mg. Pravdepodobnosť a $(x)dt$, značí že R je ďalšia reakcia, ktorá sa objaví v rámci nasledujúcej dt časovej jednotky [11].

Definícia CME:

$$\frac{d}{dt}p(x,t) = \sum_{=1}^{M} a\ (x - s\ ) \cdot p(x - s\ , t) - a\ (x) \cdot p(x,t)$$

Keďže je CME lineárna môže byť zapísaná ako $\frac{d}{dt}p(t) = p(t)Q$. Výpočet tejto rovnice si podrobne vysvetlíme v časti o Markovových reťazcoch v spojitom čase (sekcia 3.2), pretože vývoj týchto Markovových reťazcov je daný práve CME. Riešením CME je teda rozdelenie pravdepodobnosti vo všetkých stavoch Markovoveho reťazca v konkrétnom čase.

Vezmime si jednoduchý príklad s N = 3 druhy molekúl fA, B, Cg a M = 2 typy reakcií.

- R_1 A + B ! C molekula A a molekula B po spojení vytvoria molekulu C
- R_2 C ! A + B molekula C sa spätne zmení na molekulu A a B

Ak začneme s L molekulami typu A a B a so žiadnou molekulou typu C tak:

$$s(0) = \begin{bmatrix} L \\ L \\ 0 \end{bmatrix}$$

Potom stavový vektor S(t) môže nadobudnúť hodnoty

$$\begin{bmatrix} L \\ L \\ 0 \end{bmatrix}, \begin{bmatrix} L-1 \\ L-1 \\ 1 \end{bmatrix}, \begin{bmatrix} L-2 \\ L-2 \\ 2 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ L \end{bmatrix}.$$

Existuje teda $L + 1$ rôznych stavov.Veľkosť závisí od celkového počtu prítomných molekúl a od presnej formy chemických reakcií. Vo všeobecnosti CME má extrémne veľkú veľkosť, ktorá nemôže byť zvládnutá analyticky alebo numericky [12].

Kapitola 3

Markovove reťazce

Markovove reťazce patria k základným formám slúžiacim k popisu náhodných procesov. Typickým príkladom náhodného procesu môže byť hod mincou alebo kockou. Existuje široká škála oblastí, kde nájdeme praktické využitie Markovových reťazcov, či sa jedná o robotiku, informatiku, ekonomiku alebo priemyselnú výrobu.

Charakteristickou vlastnosťou Markovových reťazcov je, že sú bez pamäte. Nasledujúci stav závisí len na tom, v ktorom stave sa nachádzame práve teraz. Predchádzajúci hod kockou nemá žiaden dopad na výsledok ďalšieho hodu. Rovnako pri hode mincou máme pri každom pokuse rovnakú pravdepodobnosť, hodiť rub alebo líc bez ohľadu na predchádzajúci hod.

Markovove reťazce vieme ilustrovať pomocou konečných automatov, pri ktorých sú prechody ohodnotené pravdepodobnosťami. Rozdeľujú sa na dve základné skupiny podľa času, v ktorom prebiehajú. Prvú skupinu tvoria reťazce prebiehajúce v diskrétnom čase a druhú v spojitom čase.

Pre popis biochemických reakcií sú najčastejšie využívané Markovove reťazce v spojitom čase (angl. Continous time Markov chains), ďalej v práci uvádzané ako CTMC.

Metóda rýchlej adaptívnej uniformizácie zahŕňa diskretizáciu CTMC s ohľadom na rýchlosť prechodov. K tomu budeme potrebovať Markovove reťazce v diskrétnom čase a Poissono rozloženie. Preto si v nasledujúcej sekcii 3.1 definujeme Markovove reťazce v diskrétnom čase (angl. Discrete time Markov chains), ďalej uvádzané ako DTMC. Tieto reťazce sú intuitívnejšie a jednoduchšie na popis, ale keďže sú obmedzené na diskrétny čas vedia opísať menšie množstvo procesov. Na DTMC si jednoduchšie predstavíme základné vlastnosti a výpočty s Markovovými reťazcami[9] .

3.1 Markovove reťazce v diskrétnom čase

V tejto sekcii sa budeme venovať pravdepodobnostnému DTMC. Ako už bolo spomenuté zadefinujeme si základné vlastnosti a predstavíme príklady. Následne na základe DTMC vysvetlíme zložitejší a analyticky náročnejší CTMC.

Ešte pred samotnou definíciou Markovových reťazcov si stručne objasníme stochastické procesy. Tieto procesy sa plynutím času náhodne menia bez toho, aby sme ich vedeli kontrolovať. Jedným z typických príkladov je rast populácie baktérií. Môžeme ich definovať ako súbor náhodných premenných. Náhodným procesom môžme nazvať zobrazenie, ktoré každej hodnote k *2* N, priradí náhodnú veličinu X_k. Množina náhodných premenných

$X_1, X_2, ..., X_k, ...$ definuje stochastický proces [4].

$$fX_k : k \geq 0g$$

Markovove reťazce definujeme pomocou konečnej množiny stavov S a prechodov. Ďalej počiatočného stavu s_0, ktorý patrí do S. Každý Markovov reťazec musí spĺňať základnú Markovovu vlastnosť. Tá znie nasledovne: pravdepodobnosť, že sa dostanem do stavu s^0 v čase k + 1, je závislá len na tom, že v čase k som v stave s. Nezáleží na tom, kde som bol v predchádzajúcom čase k − 1. Definícia $P(X_k = s)$ určuje pravdepodobnosť, že proces X je v čase k v stave s.

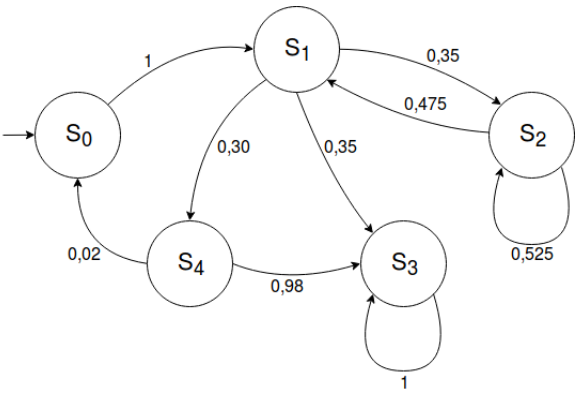
$$P(X_{k+1} = s^0 jX_k = s, X_{k-1} = s_{k-1}, ..., X_0 = s_0) = P(X_{k+1} = s^0 jX_k = s)$$

Matica pravdepodobnosti prechodu $P(s, s^0)$, udáva pravdepodobnosť vykonať prechod zo stavu s do s^0. Keďže sa jedná o pravdepodobnosť jej hodnota musí byť v rozmedzí $0 \leq P(s, s^0) \leq 1$. Súčet pravdepodobností prechodu vychádzajúcich z jedného stavu je 1.

Definícia 3.1.1 *DTMC je trojica* $D = (S, s_0, P)$ *kde:*

- S *je konečný počet stavov*
- s_0 *2* S *je počiatočný stav*
- $P : S \times S$ *!* $[0, 1]$ *je matica pravdepodobnosti prechodu kde* $\sum_{s^0 2S} P(s, s^0) = 1$ *pre všetky* s *2* S

Obrázok 3.1 nám ilustruje konečný automat popisujúci DTMC. Stavy sa znázorňujú pomocou kruhov. U biochemických systémov stavy typicky zodpovedajú počtu molekúl rôznych chemických látok. Prechody sú značené pomocou šípok s označením pravdepodobnosti prechodu, uskutočňujú sa pri vzájomnej reakcii molekúl. Počiatočný stav s_0 je značený vchádzajúcou šípkou. Zobrazený Markovov reťazec obsahuje 5 stavov $S = (s_0, s_1, s_2, s_3, s_4)$.



Obr. 3.1: Príklad Markovovho reťazca v diskrétnom čase

Daný konečný automat nám graficky znázorňuje maticu pravdepodobnosti prechodu P. V matici nám riadky znázorňujú odkiaľ vychádzajú prechody a stĺpce určujú kam. Takže prvý riadok nám určuje, že budeme vychádzať zo stavu s_0 a druhý stĺpec určuje, že pôjdeme do stavu s_1 s pravdepodobnosťou 1. Podobne pri stave s_1, 2. riadok matice, sa môžeme dostať do stavov s_2, s_3, s_4. Túto pravdepodobnosť zapíšeme na odpovedajúce miesta, teda do 2., 3. a 4. stĺpca.

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.35 & 0.35 & 0.3 \\ 0 & 0.475 & 0.525 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.02 & 0 & 0 & 0.98 & 0 \end{pmatrix}$$

3.1.2 Stavová pravdepodobnosť a tranzientná analýza

Jedna zo základných vecí, ktorá nás zaujíma pri DTMC je pravdepodobnosť že sa nachádzame v konkrétnom stave po uskutočnení určitého počtu krokov k. Teda ak začíname v stave s_0, aká je pravdepodobnosť, že sa po n krokoch stále v tomto stave nachádzame. Túto pravdepodobnosť zisťujeme pre celý reťazec stavov. Stavovú pravdepodobnosť definujeme ako $\pi_s(k) \equiv P[X_k = s]$, teda pravdepodobnosť, že v čase k sa proces X nachádza v stave s. Môžeme ďalej definovať stavový vektor

$$(k) = [\pi(k), \pi_1(k), ...]$$

Vyjadruje nám pravdepodobnosť, že sa nachádzame sa v konkrétnom stave π_j v čase k. Veľkosť vektora je určená množstvom stavov Markovovho reťazca.

Špeciálnym prípadom stavového vektora je iniciálny stavový vektor, ktorý vyjadruje pravdepodobnosť v čase 0:

$$(0) = [\pi_0(0), \pi_1(0), ...]$$

Typicky sa v čase 0 nachádzame v počiatočnom stave s_0 s pravdepodobnosťou 1. Teda ak by mal Markovov reťazec 4 stavy a 1. stav by bol počiatočný, iniciálny stavový vektor by vyzeral nasledovne

$$\pi_0 = [\ 1 \quad 0 \quad 0 \quad 0\ ]$$

Tranzientná analýza [4] sa využíva pri distribúcií pravdepodobnosti. Taktiež ju použijeme v prípade, keď budeme zisťovať pravdepodobnosť, že sa nachádzame v stave s_i v čase k. Pri výpočtoch budeme potrebovať iniciálny stavový vektor $\pi(0)$. Z ktorého sa odvíjajú výpočty.

Táto definícia nám ukazuje, že k vypočítaniu pravdepodobnosti daného stavu potrebuje len predchádzajúci stavový vektor a maticu pravdepodobnosti prechodu. Pomocou substitúcie vyjadríme vektor $(k + 2)$:

$$(k + 2) = (k + 1)P = (k)P * P$$

Ak by sme postupovali rovnako, zistili by sme, že na vyjadrenie akéhokoľvek (k) nám postačuje iniciálny stavový vektor (0):

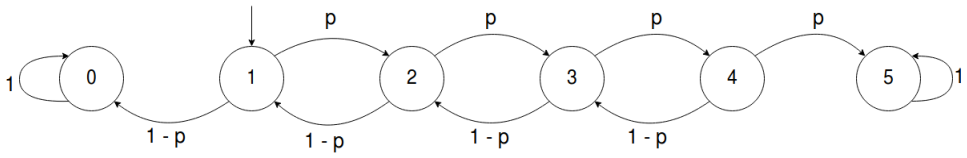
$$(k) = (0)P^k, k = 1, 2, ...$$

Vzájomné násobenie matíc je však výpočtovo náročná operácia, preto si v sekcii 4.5.2 ukážeme, ako túto metódu zrýchliť.

3.1.2.1 Problém hazardného hráča

Uvažujte o hráčovi, ktorého počiatočný majetok je 1e. Na začiatku hry vloží 1 euro a s pravdepodobnosťou p vyhrá. Pri výhre získa 1e a vráti sa mu vložené 1e. V prípade prehry

s pravdepodobnosťou p-1 stratí vložené 1e. Takto pokračuje kým sa hra neskončí. To nastane v prípade prehry všetkých peňazí alebo pri výhre 5e. Aká je pravdepodobnosť, že vyhrá 5e po 5 hrách?



Obr. 3.2: Markovov reťazec problému hazardného hráča

Vytvoríme maticu pravdepodobnosti prechodu P, ktorú ilustruje obrázok 3.2. Hru začíname v stave 1. Ak sa dostaneme do stavu 0 alebo 5 hra končí, čo vyjadrujú vlastné slučky. Teda zo stavu 1, ktorý ilustruje 2. riadok matice, sa vieme dostať s pravdepodobnosťou p do stavu 0 a s pravdepodobnosťou p-1 do stavu 2. Tieto hodnoty zapíšeme odpovedajúcim stĺpcom p_10 zapíšeme do 1. stĺpca a p_12 zapíšeme do 3. stĺpca

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1-p & 0 & p & 0 & 0 & 0 \\ 0 & 1-p & 0 & p & 0 & 0 \\ 0 & 0 & 1-p & 0 & p & 0 \\ 0 & 0 & 0 & 1-p & 0 & p \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Ako prvé si definujeme stavový vektor. Na začiatku hry je pravdepodobnosť 1, že sa nachádzame v počiatočnom stave 1 (0) = $[0, 1, 0, 0, 0, 0]$. Následne použijeme vzorec (k) = (0)P^k na vypočítanie stavu v 5. kroku

$$(5) = (0)P^5$$

Zaujíma nás pravdepodobnosť, že hráč vyhral 5e po 5 hrách, ktorá je p^4. Táto hodnota π_5 je vyjadrená ako 6. údaj vektora $\pi(5)$

$$(5) = [\ (2p(1-p)^3 + (1-p)^2)p - p + 1 \quad 0 \quad 5(1-p)^2p^3 \quad 0 \quad 3(1-p)p^4 \quad p^4\ ]$$

Pre lepšiu ilustráciu si môžeme ukázať tento príklad na konkrétnych hodnotách. Budeme počítať s pravdepodobnosťou výhry 50% teda $\frac{1}{2}$. Po dosadení hodnoty do rovnice nám vychádza, že pravdepodobnosť vyhrať hru má hráč len 0.0625 naopak pravdepodobnosť prehrať hru je 0.6875. Pravdepodobnosť, že stále hrá hru je zvyšných 0,25.

$$(5) = [\ 0,6875 \quad 0 \quad 0,15625 \quad 0 \quad 0,09375 \quad 0,0625\ ]$$

3.2 Markovove reťazce v spojitom čase

Ako sme už spomínali v úvode kapitoly chemické reakcie prebiehajú v reálnom teda spojitom čase. Analýza tohto času je náročnejšia na výpočet, pretože zmeny v systéme môžu nastať v akýkoľvek reálny čas t. Je s ním však možné popísať väčšie množstvo problémov. V DTMC sa zmeny vyskytovali iba po diskrétnych časových krokoch. Preto sú jednoduchšie na analýzu a výpočet. Je pre nás dôležité snažiť sa previesť analýzu spojitého času na

analýzu diskrétneho bez straty informácie, ktorú obsahovala. Postupne si ukážeme kroky, ktorými vieme transformovať CTMC na DTMC.

CTMC sú definované podobne ako DTMC množinou stavov S a počiatočným stavom s_0. Aj pre reťazce v spojitom čase musí platiť Markovova vlastnosť. Definuje závislosť budúceho stavu len na súčasnom podobne ako pri DTMC 3.1 lenže v spojitom čase t.

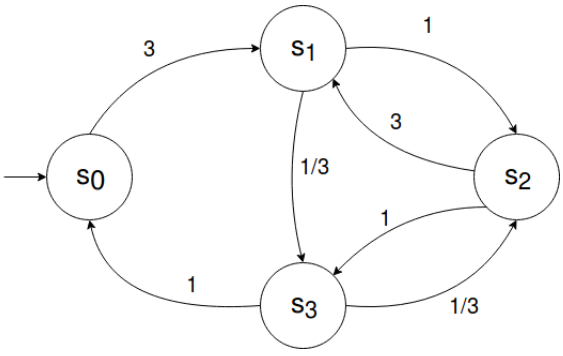
$$P(X(t_{k+1}) = s^{0} j X(t_k) = s, X(t_{k-1}) = s_{k-1}, ..., X(t_0) = s_0) = P(X(t_{k+1}) = s^{0} j X(t_k) = s)$$

Základný rozdiel medzi týmito reťazcami je, že zatiaľ čo u DTMC máme pravdepodobnosť prechodu u CTMC je to rýchlosť prechodu $E(s)$. Čím väčšia je rýchlosť tým väčšia je pravdepodobnosť prechodu. V diskrétnom čase sme využívali maticu pravdepodobnosti prechodu P. Pri spojitom čase sa však môžu udalosti vyskytnúť kedykoľvek medzi k a k + 1. Preto nemôžeme túto maticu použiť. Budeme využívať maticu prechodovej rýchlosti R, ktorá určuje rýchlosti prechodov medzi stavmi.[4]

Definícia 3.2.1 *CTMC je trojica* $C = (S, s_0, R)$ *kde:*

- S *je konečný počet stavov*
- $s_0 \epsilon S$ *je počiatočný stav*
- $R : S \times S \; ! \; R > 0$ *je matica prechodovej rýchlosti*

Ako sme už spomínali pomocou konečného automatu vieme popísať CTMC podobne ako DTMC. Jediným rozdielom je ohodnotenie hrán. Zatiaľ čo pri DTMC to boli pravdepodobnosti pri CTMC sú to rýchlosti odchodu zo stavu $E(s)$. Obrázok 3.3 nám popisuje jednoduchý príklad CTMC. Vidíme že obsahuje 4 stavy $S = (s_0, s_1, s_2, s_3)$, ďalej vieme, že počiatočný stav je s_0.



Obr. 3.3: Príklad Markovovho reťazca v spojitom čase

Matica prechodovej rýchlosti R nám schematicky vyjadruje obrázok 3.3. Táto matica sa opäť konštruuje podobne ako pravdepodobnostná matica P. Avšak zameníme pravdepodobnosti s rýchlosťami.

$$R_1 = \begin{pmatrix} 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{3} \\ 0 & 3 & 0 & 1 \\ 1 & 0 & \frac{1}{3} & 0 \end{pmatrix}$$

3.2.2 Rýchlosť odchodu

V mnohých prípadoch sa stáva, že z jedného stavu s_0 sa vieme dostať do viacerých stavov s_i. V tom prípade nemáme len jednu rýchlosť odchodu $E(s)$ zo stavu ale máme ich viac. Čas strávený v stave s_0 je exponenciálne rozdelený s rýchlosťou $E(s)$ medzi tieto stavy. Ako sa tento čas rozdeľuje si vysvetlíme v kapitole o štandardnej uniformizácii 4.5. Ak $E(s) = 0$ tak sa jedná o absorbujúci stav. Rýchlosť odchodu zo stavu s_0 je $E(s_0)$ a vypočítame ju ako sumu všetkých rýchlostí z daného stavu.

$$E(s) \stackrel{\text{def}}{=} \sum_{s' 2 S} R(s, s')$$

Pre predchádzajúcu maticu R platia rýchlosti odchodu:

- $E(s_0) = 3$
- $E(s_1) = \frac{4}{3}$
- $E(s_2) = 4$
- $E(s_3) = \frac{4}{3}$

3.2.3 Prechodová matica

Ako sme si už viackrát spomínali pri výpočtoch CTMC budeme využívať DTMC. Budeme premieňať rýchlosti na pravdepodobnosti. Využijeme k tomu prechodovú maticu, ktorá zmení rýchlosti CTMC.

Vloženie (angl. embedded) DTMC do CTMC $C = (S, s_0, R)$ je DTMC $emb(C) = (S, s_0, P^{emb(C)})$ kde pre $s, s' \ 2 \ S$:

$$P^{emb(C)}(s, s') = \begin{cases} \frac{R(s;s')}{E(s)} & ak \ E(s) \neq 0 \\ 1 & ak \ E(s) = 0 \ \ a \ s = s' \\ 0 & inak. \end{cases}$$

Pomocou definície a vypočítanej rýchlosti odchodu sme vytvorili z matice R maticu $P_1^{emb(C_1)}$. Pravdepodobnosť vychádzajúca z jedného stavu musí byť rovná 1. Zostávame v stave s po dobu meškania, ktoré je exponenciálne rozdelené s rýchlosťou $E(s)$ a následne urobíme prechod. Pravdepodobnosť že tento prechod bude do stavu s' je určená $P^{emb(C_1)}(s, s')$. [14]

$$P_1^{emb(C_1)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} \\ 0 & \frac{3}{4} & 0 & \frac{1}{4} \\ \frac{3}{4} & 0 & \frac{1}{4} & 0 \end{pmatrix}$$

3.2.4 Infinitezimálna generátorová matica

Infinitezimálna generátorová matica sa môže nazývať aj matica intenzity. Táto matica podobne ako R vyjadruje rýchlosti medzi stavmi. Oproti R je tu však zásadný rozdiel v tom, že pribudli rýchlosti odchodu zo stavu.

Pre $C = (S, s_0, R)$ je matica $Q : S \times S \ !$ definovaná ako:

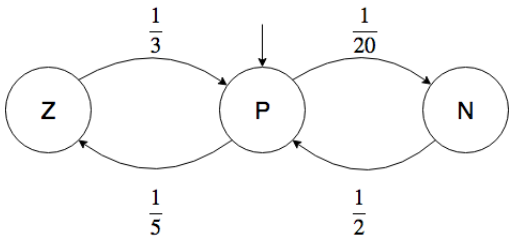
$$Q(s,s') = \begin{cases} R(s,s') & ak\ s \neq s' \\ -\sum_{s'' \neq s} R(s,s'') & inak. \end{cases}$$

Túto maticu jednoducho vytvoríme z R. Na hlavnej uhlopriečke sú vlastné slučky, ktoré znázorňujú odchádzajúcu rýchlosť. Pre túto maticu platí, že riadky sa sčítajú do 0. Túto maticu využijeme pri redukcii CTMC na DTMC [14].

$$Q_1 = \begin{pmatrix} -3 & 3 & 0 & 0 \\ 0 & -\frac{4}{3} & 1 & \frac{1}{3} \\ 0 & 3 & -4 & 1 \\ 1 & 0 & \frac{1}{3} & -\frac{4}{3} \end{pmatrix}$$

3.2.5 Príklad

V zmrzlinárni predávajú rôzne typy nanukov a zmrzlín. Ak príde k predajni zákazník a vidí, že je stánok so zmrzlinou obsadený ide si kúpiť zmrzlinu do iného stánku. Zákazníci kupujúci zmrzlinu prichádzajú do predajne v intervaloch daných Poissonovým rozdelením (sekcia 4.1) s rýchlosťou $\lambda = \frac{1}{5}$ a zákazníci kupujúci nanuk s rýchlosťou $\lambda = \frac{1}{20}$. Nabratie zmrzliny a obslúženie zákazníka trvá spolu 3 minúty $\mu_Z = \frac{1}{3}$. Ak zákazník kupuje iba nanuk trvá to len 2 minúty $\mu_Z = \frac{1}{2}$. Na tomto príklade si ilustrujeme vytvorenie matíc definovaných v predchádzajúcich podsekciách 3.2.4, 3.2.3. Budeme ich potrebovať pri nasledujúcich výpočtoch.



Obr. 3.4: Markovov reťazec predajne zmrzliny

S = [Z(zmrzlina), P(prázdna predajňa), N(nanuk)]

Rýchlosť príchodu zákazníka je rozdelená podľa tovaru, ktorý kupuje. Do prázdnej predajne nám vstupuje teda nanuk alebo zmrzlina s rôznou rýchlosťou. Rovnako po rôznej dobe z neho odchádzajú. Z daných rýchlostí vieme vytvoriť maticu prechodovej rýchlosti. Následne pomocou rýchlostí prechodu vytvoríme prechodovú maticu. Ako poslednú vytvoríme infinitezimálnu generátorovú maticu.

$$R = \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ \frac{1}{5} & 0 & \frac{1}{20} \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \quad P^{emb(C)} = \begin{pmatrix} 0 & 1 & 0 \\ \frac{4}{5} & 0 & \frac{1}{5} \\ 0 & 1 & 0 \end{pmatrix} \quad Q = \begin{pmatrix} -\frac{1}{3} & \frac{1}{3} & 0 \\ \frac{2}{10} & -\frac{1}{4} & \frac{1}{20} \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

Kapitola 4

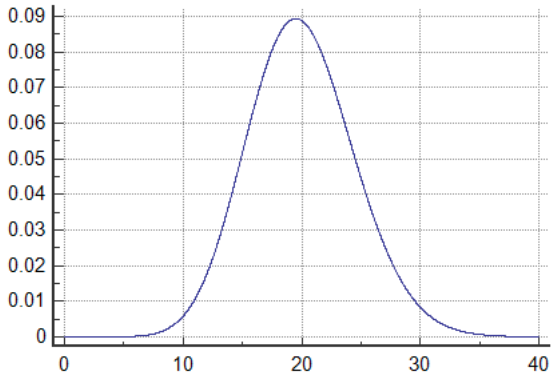
Uniformizácia

4.1 Poissonovo rozdelenie

Pri výpočte uniformizácie budeme využívať Poissonovo rozdelenie. Pre lepšie pochopenie, si ho preto definujeme ešte pred samotnou uniformizáciou. Poissonovo rozdelenie sa používa na skúmanie výskytu istého javu. Napríklad príchod zákazníka do obchodu alebo vznik chybného kusu pri výrobe. Náhodná veličina X, udáva počet udalostí, ktoré nastanú za jednotku času. Vieme, že priemerne nastáva λ udalostí za danú jednotku času. Pravdepodobnostná funkcia Poissonovho rozdelenia s parametrom λ je

$$\gamma(X = i) = \frac{\lambda^i}{i!} e^{-} \quad , i = 0, 1, 2, ...$$

Zanedbateľné hodnoty sa nachádzajú na začiatku a konci Poissonovho rozloženia. Napríklad pri rýchlosti $\lambda = 20$ vidíme (obr. 4.1), že od 0 do 5 je pravdepodobnosť bezvýznamná rovnako aj od 35 ďalej. Pri vyšších číslach je zanedbateľných stavov omnoho viac. V nasledujúcej kapitole si ukážeme spôsob ako obmedziť výpočet tohto rozloženia len na stavy s podstatnou pravdepodobnosťou.



Obr. 4.1: Poissonovo rozloženie [1]

[1] https://www.medcalc.org/manual/poisson_distribution_functions.php

4.2 Fox Glynnov algoritmus

Vo všeobecnosti sa dá tento algoritmus využiť pri výpočtoch zahŕňajúcich Poissonove pravdepodobnosti. Predstavíme si zjednodušenú verziu algoritmu, keďže jeho implementáciu len využívame. Detailné vysvetlenie tohto algoritmu sa nachádza v článku Computing Poisson Probabilities [7]. Tento algoritmus využijeme pri výpočte štandardnej aj adaptívnej uniformizácie v nasledujúcich sekciách 4.5 4.6.

V podstate nám tento algoritmus nájde pravý R a ľavý L bod skrátenia $\sum_{i=L}^{R} \frac{i}{i!}e^{-} \leq 1 - \epsilon$. K tomu budeme potrebovať hodnotu ϵ, ktorá nám určuje presnosť výpočtu. Fox Glynnov algoritmus nám zaručuje, že pravdepodobnosť medzi bodmi L a R je väčšia ako $1 - \epsilon$. Zvyšná pravdepodobnosť mimo bodov skrátenia teda bude menšia ako ϵ. Najjednoduchší spôsob výpočtu je postupne zvyšovať hornú hranicu i :

$$\sum_{i=0}^{k} \frac{\lambda^{i}}{i!}e^{-} \leq 1 - \epsilon$$

V bode dosiahnutia dostatočnej presnosti ϵ ukončíme výpočet, čo nám ušetrí množstvo počítania. Tento jednoduchý spôsob však nerieši problém počiatočných zanedbateľných hodnôt. V prípade ak bude $\lambda = 1000$ je množstvo začiatočných zanedbateľných stavov 900.

4.3 Výpočet skrátenia

Vieme určiť pravdepodobnosť, že sa udeje presne i udalostí z definície Poissonovho rozloženia $p(i) = \frac{i}{i!}e^{-}$. Pri definovaní výpočtu pravého bodu skrátenia budeme počítať, že sa udialo aspoň i udalostí $Q\ (i) = \sum_{j=i}^{1} p\ (j)$. Naopak pri výpočte ľavého bodu skrátenia budeme počítať pravdepodobnosť, že sa udialo najviac i udalostí $T\ (i) = \sum_{j=0}^{i} p\ (j)$.

K nasledujúcim výpočtom budeme potrebovať určiť premenné

$$\begin{aligned} a &= (1 + \tfrac{1}{}) e^{\frac{1}{16}} p_{\overline{2}} \\ b &= (1 + \tfrac{1}{}) e^{\frac{1}{8}} \\ d(k, \lambda) &= \frac{1}{\left(1 - exp\left(-\frac{2}{9}\left(k \sqrt{2} + \frac{3}{2}\right)\right)\right)} \\ modus\ m &= b\lambda c \end{aligned}$$

K výpočtu ľavého bodu skrátenia L využijeme definíciu pre $T\ (i)$. Keďže budeme počítať 2 body skrátenia, pravdepodobnosť od pravého a ľavého bodu bude postačujúca, ak bude nanajvýš $\frac{}{2}$.

Definícia 4.3.1 *Ak* $\lambda \geq 2$ *a* $k \geq \frac{1}{2}$

$$T\left(\left\lfloor m - k \sqrt{\lambda} - \frac{3}{2} \right\rfloor\right) \leq \frac{b\ e^{\frac{-k^2}{2}}}{k \sqrt{2\pi}}$$

Ľavý bod skrátenia L vypočítame tak, že vezmeme spodnú hranicu k a postupne ju zväčšujeme o 1 pokiaľ nieje rovnica menšia ako $\frac{}{2}$.

$$T\left(\left\lfloor m - k\sqrt{\lambda} - \frac{3}{2}\right\rfloor\right) \leq \frac{b\ e^{\frac{-k^2}{2}}}{k\sqrt{2\pi}} \leq \frac{}{2}$$

Tak potom ľavý bod skrátenia vypočítame ako $L = \left\lfloor m - k\sqrt{\lambda} - \frac{3}{2}\right\rfloor$

Definícia 4.3.2 *Ak* $\lambda \geq 2$ *a* $\frac{1}{2\sqrt{2}} \leq k \leq \frac{}{2\sqrt{2}}$

$$Q\left(\left\lceil m + k\sqrt{2\lambda} + \frac{3}{2}\right\rceil\right) \leq \frac{a\ d(k,\lambda)e^{\frac{-k^2}{2}}}{k\sqrt{2\pi}}$$

Pravý bod skrátenia R vypočítame tak, že vezmeme spodnú hranicu k a postupne ju zväčšujeme o 1 pokiaľ nieje rovnica menšia ako $\frac{}{2}$.

$$Q\left(\left\lceil m + k\sqrt{2\lambda} + \frac{3}{2}\right\rceil\right) \leq \frac{a\ d(k,)e^{\frac{-k^2}{2}}}{k\sqrt{2}} \leq \frac{}{2}$$

Tak potom pravý bod skrátenia vypočítame ako $R = \left\lceil m + k\sqrt{2\lambda} + \frac{3}{2}\right\rceil$

4.3.3 Príklad

K výpočtu Fox Glynnovho algoritmu potrebujeme určiť $\lambda = 300$ a $\epsilon = 1 * 10^{-}8$. Keďže je splnená podmienka 300 ≥ 2. Zvolíme *k* aby bola splnená aj nasledujúca $k \geq \frac{1}{2*300}$ Pre prvý výpočet zvolíme *k*=1.

$$\frac{(1 + \frac{1}{300})e^{\frac{1}{8*300}}e^{\frac{-1^2}{2}}}{1 * \sqrt{2\pi}} = 0,24$$

Postupne zväčšujeme *k* kým výsledok nieje menší ako $5 * 10^{-9}$

$$\frac{(1 + \frac{1}{300})e^{\frac{1}{8*300}}e^{\frac{-6^2}{2}}}{6 * \sqrt{2\pi}} = 1 * 10^{-}9$$

Postupne sme sa dopracovali k výsledku *k* = 6.

$$T_{300}\left(\left\lfloor 300 - 6\sqrt{300} - \frac{3}{2}\right\rfloor\right) \leq 5 * 10^{-9}$$

Tento výpočet nám určil L = 194.

Podobne ako pri ľavom bode aj pri pravom bode zvolíme premennú *k* spĺňajúcu podmienku $\frac{1}{2\sqrt{2*300}} \leq k$

$$\frac{(1 + \frac{1}{300})e^{\frac{1}{16}}\sqrt{2}\frac{1}{\left(1-exp\left(-\frac{2}{9}\left(1\sqrt{2*300}+\frac{3}{2}\right)\right)\right)}e^{\frac{-1^2}{2}}}{1\sqrt{2\pi}} = 0,27$$

Postupne zväčšujeme *k*, kým výsledok nieje menší ako $5 * 10^{-9}$ a zároveň musí platiť, aby $k \leq \frac{\sqrt{300}}{2\sqrt{2}}$.

$$\frac{(1+\frac{1}{300})e^{\frac{1}{16}}\sqrt{2}\frac{1}{\left(1-exp\left(-\frac{2}{9}\left(6\sqrt{2*300}+\frac{3}{2}\right)\right)\right)}e^{\frac{-6^2}{2}}}{6\sqrt{2\pi}} = 1,1 * 10^{-9}$$

Výsledná hodnota nám vyšla $k = 6$.

$$Q_{300}(\lceil 300 + 6\sqrt{2 * 300} + \frac{3}{2}\rceil) \leq 5 * 10^{-9}$$

Výsledné R = 449.

Tento výpočet nám zredukoval $\sum_{k=0}^{k} \frac{300^k}{k!}e^{-300} \leq 1 - \epsilon$ na $\sum_{k=179}^{472} \frac{300^k}{k!}e^{-300}$. Namiesto počítania 179 stavov so zanedbateľnou pravdepodobnosťou sme pomocou 12 výpočtov určili hranice výpočtu.

4.4 Výpočet váh

V momente, keď už vieme vypočítať body skrátenia môžeme prejsť na výpočet diskrétneho Poissonovho rozloženia. Je dôležitý pre efektívnosť a správnosť uniformizačného algoritmu.

$$\gamma_t(k) = \frac{(\lambda t)^k}{k!}e^{-t}$$

Počet váh, ktoré budeme počítať je R - L , teda $w[i], L < i < R$. Tieto hodnoty počítame rekurzívne od hodnoty modusu m = $b\lambda tc$ po body skrátenia $w(m - 1), w(m - 2), ..., w(L)$ a $w(m + 1), w(m + 2), ..., w(R)$. Od iniciálnej hodnoty modusu $w(m) = \frac{\Omega}{10^{10}(R-L)}$ budeme počítať jej predchádzajúce a nasledovné hodnoty. Pričom je rovná najväčšiemu možnému číslu. Nazýva sa tiež prah pretečenia. Predchádzajúce váhy vypočítame ako $w(j - 1) = \frac{j}{j+1}w(j), j = m$. Následníkov $w(m)$ vypočítame ako $w(j + 1) = \frac{j}{}w(j), j = m$. Pre váhy platí, že ich suma sa rovná 1 $\sum_i w[i] = 1$. Celková váha W sa rovná súčtu všetkých váh $W = w(L) + ... + w(R)$ [3].

Diskrétne pravdepodobnosti sa rovnajú:

$$\gamma_t(k) = w[k]/W$$

Týmto výpočtom sme značne urýchlili a zjednodušili výpočet Poissonovho rozloženia.

4.5 Štandardná uniformizácia

Predstavíme si techniku štandardnej uniformizácie (angl. standart uniformization), ďalej uvádzaná ako SU. Táto metóda je tiež známa ako metóda randomizácie alebo Jensenova metóda. Využíva sa pri pravdepodobnostných metódach, slúži na výpočet prechodových riešení konečných CTMC, pomocou aproximácie procesu DTMC.

Pôvodný Markovov reťazec modifikujeme pomocou najväčšej prechodovej rýchlosti $\lambda \geq max\{E(s)|s \in S\}$, ktorá sa nazýva aj uniformizačná rýchlosť. Všetky rýchlosti v CTMC sú normalizované vzhľadom na λ. To znamená, že každý stav s s rýchlosťou $E(s) = \lambda$, v jednej epoche uniformizovanej matice zodpovedá jednému exponenciónalne rozdelenému meškaniu s rýchlosťou λ. Pri týchto stavoch nevzniká žiadna dodatočná vlastná slučka. V prípade ak je stav s pomalší $E(s) < \lambda$, má dlhšiu dobu pobytu. Jedna epocha nieje dostatočne dlhá na uskutočnenie prechodu. Preto sa tieto stavy môžu vrátiť späť do seba samého pomocou vlastnej slučky s pravdepodobnosťou 1 – $\frac{E(s)}{}$.

Definícia 4.5.1 *Pre všetky CTMC* $C = (S, s_0, R)$ *s infinitezimálnou generátorovou maticou* Q*, a uniformizačnou rýchlosťou* $\lambda \geq max\{E(s) | s \in S\}$*, uniformizované DTMC je dané* $unif(C) = (S, s_0, P^{unif(C)})$ *kde:*

$$P^{unif(C)}(D(k+1) = s_j | D(k) = s_i) = \begin{cases} \frac{Q(s_i,s_j)}{\lambda} & ak\ s_i \neq s_j \\ 1 - \frac{-i}{\lambda} & ak\ s_i = s_j. \end{cases}$$

Maticový zápis:

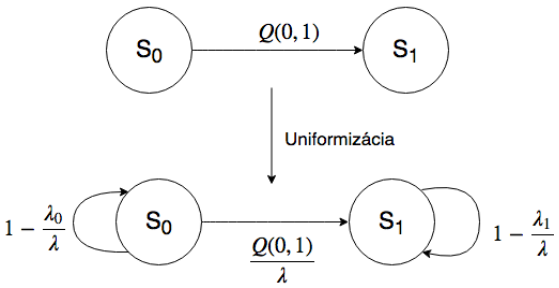
$$P^{unif(C)} = I + \frac{Q}{\lambda}$$

Pomocou týchto definícii pretransformujeme infinitezimálnu generátorovú maticu Q_1 zo sekcie 3.2.4 na uniformizačnú maticu. Jednotková matica I má veľkosť n. Je rozmeru štvorca $n \times n$. Na hlavnej diagonále sa nachádzajú 1 na všetkých ostatných miestach sú 0. Parameter uniformizovanej rýchlosti λ sme vypočítali z rýchlostí odchodu uvedených v sekcii 3.2.2 a rovná sa 4.

$$Q_1 = \begin{pmatrix} -3 & 3 & 0 & 0 \\ 0 & -\frac{4}{3} & 1 & \frac{1}{3} \\ 0 & 3 & -4 & 1 \\ 1 & 0 & \frac{1}{3} & -\frac{4}{3} \end{pmatrix} \quad P_1^{unif(C)} = \begin{pmatrix} \frac{1}{4} & \frac{3}{4} & 0 & 0 \\ 0 & \frac{8}{12} & \frac{3}{12} & \frac{1}{12} \\ 0 & \frac{3}{4} & 0 & \frac{1}{4} \\ \frac{3}{12} & 0 & \frac{1}{12} & \frac{8}{12} \end{pmatrix}$$

Matica $P_1^{unif(C)}$ opäť spĺňa pravidlo, že riadky sa sčítajú do 1. Pri stave $Q_1(2,2)$ je v matici $P_1^{unif(C)}$ vidieť, že rýchlosť sa premenila na $P_1^{unif(C)}(2,2) = 0$. Keďže hodnota λ sa rovnala hodnote $E(2)$ znamená to, že v daný bod nemá vlastnú slučku.

Keď už sme mali možnosť pracovať s uniformizovanou maticou, môžeme si všimnúť rozdiel oproti vloženému DTMC $P^{emb(C)}$. Epocha sa pri CTMC a $emb(C)$ zhoduje. Pri uniformizácii $unif(C)$ epocha zodpovedá jedinému exponenciónalnemu rozloženému oneskoreniu s rýchlosťou λ v C.



Obr. 4.2: Grafické znázornenie uniformizácie

Z obrázku 4.2 vidíme, že pri uniformizácii vznikajú nové vlastné slučky. Na prvý pohľad by sa mohlo zdať že sú tieto stavy navyše. Tieto vlastné slučky nemajú žiaden vplyv na chovanie Markovovho reťazca. Daný stav nebol ovplyvnený a keďže pri Markových modeloch platí že nezáleží na predchádzajúcom stave čas, ktorý zostáva do ďalšieho prechodu nebol ovplyvnený, naďalej bude exponenciálne rozdelený s rýchlosťou λ_i.

Definujeme si maticu všetkých prechodových pravdepodobností pre čas t, $\mathbf{\Pi}_t(s_i, s_j) = \pi(s_j)$. S použitím uniformizovného DTMC môže byť vyjadrená ako:

$$\mathbf{\Pi}_t = \sum_{k=0}^{1} \gamma_k(\lambda t) \cdot \left(P^{unif(C)}\right)^k \; kde \; \gamma_k(\lambda t) = \frac{(\lambda t)^k}{k!} e^{-\;t}$$

Pre začiatočnú distribúciu stavového vektora $\pi(0)$, v čase t, vypočítame $\pi(t)$ ako

$$\pi(t) = \pi(0) \cdot \mathbf{\Pi}_t = \pi(0) \cdot \sum_{k=0}^{1} \gamma_k(\lambda t) \cdot \left(P^{unif(C)}\right)^k$$

Vyjadruje nám rovnicu pre výpočet stavového vektora (podsekcia 3.1.2) v čase (t). Ak sa na to pozrieme lepšie, môžeme si uvedomiť, že jeden krok uniformizovaného DTMC zodpovedá jednému exponencionálne rozloženému meškaniu s parametrom λ. Mocnina matice $\left(P^{unif(C)}\right)^k$ nám dáva pravdepodobnosť prechodu medzi všetkými dvojicami stavov v DTMC v k krokoch. Poissonove rozloženie $\gamma_k(\lambda t)$ s parametrom λt je pravdepodobnosť, že tieto prechody sa uskutočnia v čase t, vzhľadom na meškanie rozdelené s rýchlosťou λ.

V definícii vektora $\pi(t)$ sa však teraz nachádza nekonečná suma. Na skrátenie využijeme techniku Fox Glynn (sekcia 4.2), ktorú sme si už definovali. Umožní nám efektívne vypočítať $\gamma(\lambda t)$ ale aj hranice L a R.

4.5.2 Tranzientná analýza

V prípade tranzientnej analýzy, ktorú sme si už spomínali v sekcii 3.1.2, násobíme maticu s maticou. Tento matematický úkon je výpočtovo náročný a preto sa ho budeme snažiť zefektívniť. Preusporiadaním môžeme vyjadriť stavový vektor ako sumu vektorov namiesto sumy mocniny matice.

$$\pi(t) = \sum_{k=0}^{1} \gamma_k(\lambda t) \cdot \pi(0) \cdot \left(P^{unif(C)}\right)^k$$

Stavový vektor v každom prvku súčtu je vypočítaný násobením matice a vektora pomocou vektora z predchádzajúcej iterácie.

$$\pi(0)\left(P^{unif(C)}\right)^k = \left(\pi(0) \cdot \left(P^{unif(C)}\right)^{k-1}\right) \cdot P^{unif(C)}$$

Nahradili sme násobenie matice s maticou násobením vektora s maticou,

4.6 Adaptívna uniformizácia

Rozdiel medzi štandardnou a adaptívnou uniformizáciou je v počítaní uniformizačnej matice. Štandardnú uniformizačnú maticu vypočítame na začiatku výpočtu s maximálnou prechodovou rýchlosťou λ. Pri adaptívnej uniformizácií túto maximálnu rýchlosť môžeme zmeniť pri každom kroku procesu, takže nemáme jednu rýchlosť λ, ale máme sekvenciu rýchlostí $\lambda_0, \lambda_1, \lambda_2, \ldots$ *2* $\mathsf{R}_{>0}$ [19].

$$P_k^{unif(C)}(D(k+1) = s_j | D(k) = s_i) = \begin{cases} \frac{Q(s_i,s_j)}{k} & ak \; s_i \neq s_j \\ 1 - \frac{i}{k} & ak \; s_i = s_j. \end{cases}$$

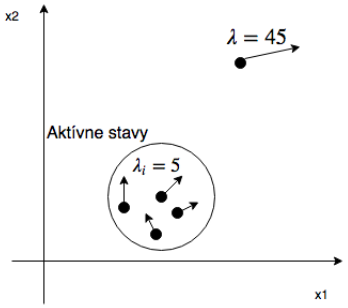
Maticový zápis:

$$P_k^{unif(C)} = I + \frac{1}{\lambda_k} Q_k, \; k = 0, 1, 2, \ldots$$

Lokálnu maximálnu rýchlosť λ_i vypočítame z aktívnych stavov. Pod týmito stavmi rozumieme tie, ktoré majú nenulovú hodnotu v stavovom vektore $\pi(k)$. Táto hodnota znamená, že existuje možnosť nachádzania sa v danom stave.

$$A_n = fs_i \; 2 \; Sj\pi_n(s_i) > 0g, A_n \quad S$$

Zásadná výhoda v porovnaní so štandardnou unifomizáciu je v prípade, ak maximálna rýchlosť λ sa nenachádza medzi aktívnymi stavmi. Teda všetky stavy sú pomalšie. To znamená, že za jeden krok môžeme prejsť menšiu vzdialenosť. V niektorých prípadoch sa za dobu behu výpočtu ani nemusíme dostať k najrýchlejšiemu stavu.



Obr. 4.3: Ukážka stavového priestoru

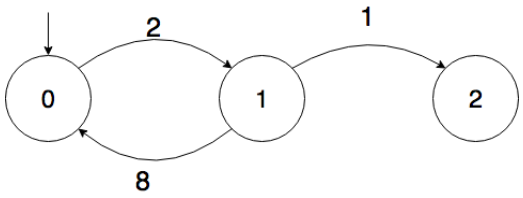
Na obrázku 4.3 môžeme jasne vidieť, že síce maximálna rýchlosť λ je 45. Avšak táto rýchlosť sa nenachádza medzi aktívnymi stavmi, takže nieje možné spraviť, taký rýchli prechod. Preto bude postačujúca lokálna maximálna rýchlosť $\lambda_i = 5$.

4.7 Príklad

Pre názornú ukážku rozdielu medzi štandardnou a adaptívnou uniformizáciou vypočítame ten istý príklad obidvoma spôsobmi. Uvažujme o opravárovi, ktorý opravuje stroj skladajúci sa z dvoch častí. Stroj je opraviteľný pokiaľ funguje aspoň jedna súčiastka. V momente, keď sa pokazia všetky súčiastky, stáva sa stroj ďalej nepoužiteľný. Rýchlosť opravy je podstatne väčšia ako rýchlosť pokazenia sa. Prvá časť sa kazí s rýchlosťou 2 a následne sa druhá kazí s rýchlosťou 1. Oprava má rýchlosť 8. Zo zadaných údajov vytvoríme maticu prechodovej rýchlosti R.

$$R = \begin{pmatrix} 0 & 2 & 0 \\ 8 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Pre lepšiu ilustráciu si vytvoríme grafické znázornenie príkladu pozri obrázok 4.4. Začíname s nepokazenými súčiastkami takže v stave nula. $S = 0, 1, 2$ čísla znázorňujú množstvo pokazených súčiastok.



Obr. 4.4: Markovov reťazec stroja

4.7.1 Riešenie štandardnou uniformizáciou

Prvý krok je vypočítanie infinitezimálnej generátorovej matice Q. Následne zistíme maximálnu prechodovú rýchlosť λ. Pomocou týchto hodnôt vytvoríme uniformizačnú maticu $P^{unif(C)}$, ktorá sa nebude meniť počas celého výpočtu.

$$Q = \begin{pmatrix} -2 & 2 & 0 \\ 8 & -9 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \ \lambda = 9, P^{unif(C)} = \begin{pmatrix} \frac{7}{9} & \frac{2}{9} & 0 \\ \frac{8}{9} & 0 & \frac{1}{9} \\ 0 & 0 & 1 \end{pmatrix}$$

Stavové vektory nám určujú ako sa bude vyzerať pravdepodobnosť po 2 krokoch bez zmenenia rýchlosti. Ako sme si definovali v sekcii4.5.2 už nemusíme násobiť navzájom matice, stačí ak vždy použijeme predchádzajúci stavový vektor.

$$\begin{aligned} \pi_0 &= (1,0,0) \\ \pi_1 &= \pi_0 \cdot P^{unif(C)} = (\tfrac{7}{9}, \tfrac{2}{9}, 0) \\ \pi_2 &= \pi_1 \cdot P^{unif(C)} = (\tfrac{65}{81}, \tfrac{14}{81}, \tfrac{2}{81}) \end{aligned}$$

4.7.1.1 Riešenie adaptívnou uniformizáciou

V tomto prípade vypočítame tiež maticu Q, avšak maximálnu rýchlosť prechodu λ zisťujeme iba z aktívnych stavov. Zo stavového vektoru $\pi_0 = (1,0,0)$ zmapujeme aktívne stavy. V našom prípade má nenulovú hodnotu iba stav 1, teda $A_0 = \{0\}$

$$Q_0 = \begin{pmatrix} -2 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ \lambda_0 = 2, P_0^{unif(C)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Keďže máme aktívny iba prvý stav prispôsobíme tomu maticu Q. Rovnako aj maximálna lokálna rýchlosť bude počítaná iba z nenulových stavov. Následne vytvoríme uniformizačnú maticu. Pomocou tranzientnej analýzy teda násobenia vektora π_0 a matice $P_0^{unif(C)}$ dostaneme nasledujúci vektor π_1, ktorý nám hovorí, že v druhom kroku sa budeme nachádzať v stave 1. Z vektoru π_1 odvodíme všetky hodnoty podobne, ako v predchádzajúcom príklade.

$$\pi_1 = (0,1,0), A_1 = \{1\}, Q_1 = \begin{pmatrix} 0 & 0 & 0 \\ 8 & -9 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \ \lambda_1 = 9, P_1^{unif(C)} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{8}{9} & 0 & \frac{1}{9} \\ 0 & 0 & 1 \end{pmatrix}$$

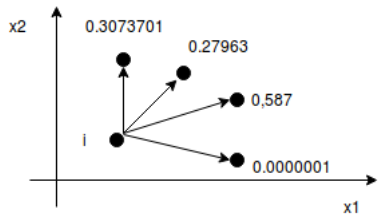
$$\pi_2 = \left(\frac{8}{9}, 0, \frac{1}{9}\right), A_2 = \{0,2\}, Q_2 = \begin{pmatrix} -2 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ \lambda_2 = 2, P_2^{unif(C)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Po výpočte 2 krokov obidvoma uniformizáciami vidíme, že pravdepodobnosť je rozdistribuovaná rozdielne.

Kapitola 5

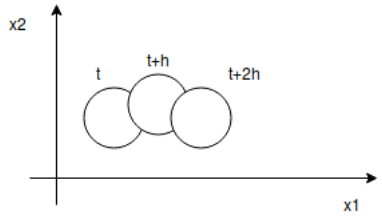
Rýchla adaptívna uniformizácia

Metóda vychádza z dvoch základných predpokladov. Prvý znie nasledovne, keď sa do niektorého stavu dostaneme s veľmi malou pravdepodobnosťou, môžeme tento stav do budúcnosti zanedbať. Podľa toho čo budeme považovať za malú pravdepodobnosť, nám samozrejme na druhú stranu bude vznikať chybovosť, s ktorou musíme počítať. Preto je dôležité správne zvoliť bod prahu, od ktorého budeme považovať pravdepodobnosť za zanedbateľnú.



Obr. 5.1: Šírenie pravdepodobnosti v stavovom priestore

Ďalší fakt z ktorého táto metóda vychádza je ten, že väčšinu času nie sme v celom stavovom priestore, ale iba v jeho časti. V praxi to znamená, že nemusíme počítať so všetkými stavmi, pretože väčšinu času sa počítame iba v malej časti priestoru. Zvyšná väčšia časť je nulová. O túto časť sa už postarala adaptívna uniformizácia, ktorá počíta iba s aktívnymi stavmi.



Obr. 5.2: Posun v čase v stavovom priestore

Rýchlu adaptívnu uniformizáciu (angl. fast adaptive uniformisation) budeme v ďalej texte označovať ako FAU. Základnou myšlienkou FAU je rozdeliť Markovov reťazec v spojitom čase $fs(t), t\epsilon R_{\geq}0g$ na dva samostatné náhodné procesy. Prvý proces je Markovov reťazec v diskrétnom čase $fD(k), k\epsilon Ng$, uchováva nám informácie o stave. Druhý je Markovov reťazec v spojitom čase, ktorý sa chová ako jednoduché počítadlo diskrétnych krokov

v spojitom čase $fB(t), t\epsilon R_{\geq}0g$. Nazýva sa proces narodenia (angl. birth process) a určuje nám plynutie času. [16]

5.1 Abstrakcia s prahom

Základná myšlienka abstrakcie je odfiltrovanie zanedbateľných stavov. Pravdepodobnosť že, sa dostaneme v danej dobe do týchto stavov je bezvýznamná. K aproximácií budeme potrebovať malý kladný prah δ, pomocou ktorého budeme meniť vektor $\pi^{(0)}$ na $\hat{\pi}^{(0)}$. Budeme prechádzať cez celý vektor π_i a všetky hodnoty, ktoré sú menšie ako δ zmeníme na 0. Symbol ^ znamená abstrakciu s prahom. Touto aproximáciou znížime počet aktívnych stavov (sekcia 4.6).

$$P(D(k+1) = \hat{s}) \; ' \; \hat{\pi}^{(k+1)}(\hat{s}) = \sum_{s;\hat{\pi}^{(k)}(s)\geq} \hat{\pi}^{(k)}(s) P(D(k+1) = \hat{s} | D(k) = s)$$

V stavovom priestore sa v každom kroku nachádzajú oblasti, v ktorých sa hromadí pravdepodobnosť. V týchto oblastiach existuje relatívne malý počet stavov s pomerne vysokou pravdepodobnosťou. Čím viac sa stav od týchto stavov vzďaľuje, tým viac mu klesá pravdepodobnosť.

Pri zmenšení počtu aktívnych stavov môže adaptívna uniformizácia pracovať s menšou hodnotou λ_k, pretože maximálna rýchlosť λ_k z aktívnych stavov pred aproximáciou môže byť väčšia ako po nej $\hat{\lambda}_k$.

$$\lambda_k = sup_{s;\pi^k(s)>0}\lambda_s \geq \hat{\lambda}_k = sup_{s;\hat{\pi}^k(s)>0}\lambda_s$$

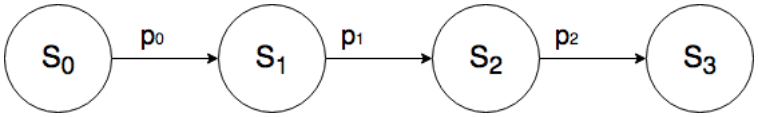
Pri našich experimentoch sa pohybujeme s prahom δ v rozmedzí $10^{-9} \leq \delta \leq 10^{-15}$. Výsledné hodnoty sú s touto aproximáciou dostatočne presné. Zároveň dosahuje značné úspory pri sieťach opisujúcich chemické rekcie.

5.2 Proces narodenia

Pri štandardnej uniformizácií je nositeľom času Poissonovo rozdelenie. Uniformizácia prebiehala pomocou maximálnej rýchlosti λ. Proces narodenia $PfB(t) = kg$ je špeciálny prípad Poissonovho procesu s konštantnou rýchlosťou = $(\lambda, \lambda, ...)$. Pri adaptívnej uniformizácii pre proces narodenia v čase t neexistuje také jednoduché riešenie. Rýchlosť sa môže zmeniť s každým krokom = $(\lambda_0, \lambda_1, ...)$. Z toho vyplýva, že nemôžeme použiť tento proces. [9]

$$\pi(t) \; ' \; \sum_{k=L}^{R} \pi(k) P(B(t) = k), \; P(B(t) = k) = \gamma_k(\lambda t) = e^{-t} \frac{(\lambda t)^k}{k!}$$

Proces B je špeciálny prípad Markovovho reťazca v spojitom čase, kde je iba jeden typ stavových prechodov a to zväčšenie stavu o 1. Tento Markovov reťazec si najskôr definujeme v jednoduchšej diskrétnej verzii. V tomto prípade sa stav s_i zvyšuje o 1 s pravdepodobnosťou p. Prechod do stavu s_{i+1} sa nazýva proces narodenia.



Obr. 5.3: Markovov reťazec procesu narodenia v diskrétnom čase

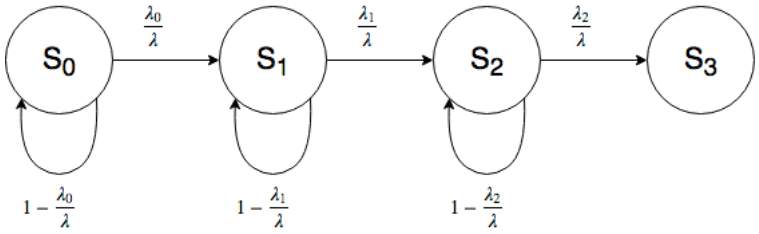
Ďalej môžeme skombinovať štandardnú uniformizáciu a abstrakciu s prahom.

$$P(B(t) = k) \approx \sum_{l=L}^{R} w_k^B(l)\gamma_l(\lambda t)$$

Aproximácia stavu pravdepodobnosti DTMC D_B spojená s B sa značí $w_{s_k}^B(l+1)$. Ešte pred samotným výpočtom si potrebujeme definovať výpočet pravdepodobností prechodu, keďže sa mení rýchlosť λ.

$$p_{s_k;s_j} = P(D^B(l+1) = s_j | D^B(l) = s_k) = \begin{cases} 1 - \hat{\lambda}_k & ak\ s_j = s_k \\ \hat{\lambda}_k & ak\ s_j = s_k + 1 \\ 0 & inak \end{cases}$$

Grafické znázornenie procesu narodenia v spojitom čase znázornené na obrázku 5.4.

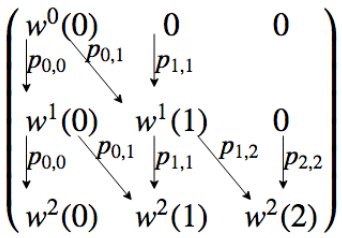


Obr. 5.4: Markovov reťazec procesu narodenia v spojitom čase

Výpočet D_B:

$$w_{s_k}^B(l+1) = w_{s_k}^B(l) * p_{s_k;s_k} + w_{s_{k-1}}^B(l) * p_{s_{k-1};s_k}$$

Maticové znázornenie výpočtu



Obr. 5.5: Matica výpočtu procesu narodenia

Nad hlavnou diagonálou sú na obrázku 5.5 stavy s pravdepodobnosťou 0. Ak vezmeme prvý riadok matice vidíme, že iba na prvom mieste je nenulová pravdepodobnostná hodnota.

V prvok kroku sa však môžeme nachádzať iba v 1. stave, takže táto hodnota sa bude rovnať 1. Obdobne v 2. kroku sa môžeme nachádzať iba v prvých dvoch stavoch, teda iba tieto dva stavy budú nenulové.

Výsledný výpočet stavového vektora $\pi(t)$ pomocou procesu narodenia.

$$\pi(t) \; ' \; \sum_{k=0}^{R} \mathfrak{v}(k) \cdot \sum_{l=L}^{R} w_k^B(l) \cdot \gamma_l(\lambda t) =: \mathfrak{n}(t)$$

Pri výpočte budeme hodnotu $w_k^B(l) \cdot \gamma_l(\lambda t)$ značiť $\beta(k)$. Táto hodnota nám určuje pravdepodobnosť, že urobíme presne k krokov v danom čase.

Obidve skrátenia vedú k viacerým aproximáciam vektoru $\pi(t)$. To isté robí aj abstrakcia s prahom pri vektore $\mathfrak{v}(k)$. Takže aproximácia $\mathfrak{p}(t)$ je viacnásobnou aproximáciou $p(t)$. Celková chyba je daná $1 - \sum_{(s2S)} \mathfrak{p}_x(t)$.

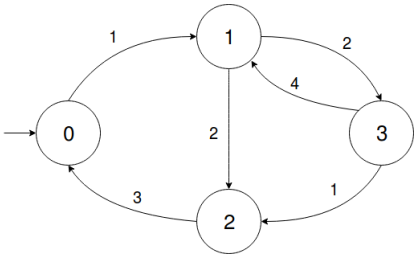
Tento výpočet budeme opakovať do bodu skrátenia R alebo pokiaľ $\beta < \epsilon, \beta = \sum_{l=L}^{R} \beta(l)$. Preto pridávame pravdepodobnosť narodenia pri každom kroku a zastavíme prechod stavovým priestorom, ak jeho hodnota získa aspoň nami určenú presnosť ϵ. Na rozdiel od štandardnej uniformizácie, kde bol bod skrátenia pevne určený,teraz vieme rozhodnúť počas výpočtu, kedy môže byť suma bezpečne skrátená.

5.3 Príklad

Zadaním príkladu bude v tomto prípade matica prechodovej rýchlosti R a počiatočný stav bude 0. Chceme zistiť aké bude výsledné pravdepodobnostné rozloženie. Zvolíme si presnosť $\epsilon = 0,9$ a prah $\delta = 0,1$.

$$R = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 3 & 0 & 0 & 0 \\ 0 & 4 & 1 & 0 \end{pmatrix}$$

Pre lepšiu predstavu si uvedieme aj ilustráciu 5.6 matice prechodovej rýchlosti R. Zistíme si maximálnu rýchlosť prechodu $\lambda = 5$.



Obr. 5.6: Markovov reťazec v spojitom čase

Prvým krokom, ktorý spravíme bude využitie Fox-glynovho algoritmu na zistenie ľavého a pravého ohraničenia. Ďalšou hodnotou, ktorú ním zistíme budú váhy.

$$\gamma[\ 0,2231 \quad 0,3347 \quad 0,251 \quad 0,1255\ ],\ L = 0, R = 3$$

Náš výpočet budeme budeme opakovať 4 krát. Ak by však ľavé ohraničenie bolo väčšie ako 0 musíme vypočítať všetky hodnoty stavového vektora aj pred hodnotou L. Násobiť váhami však začneme až keď dosiahneme hranicu L.

Vieme že počiatočný stav je 0, preto iniciálny stavový vektor bude vyzerať nasledovne w_0 = [1 0 0 0]. Skontrolujeme či niektorý zo stavov nemá menšiu pravdepodobnosť ako prah δ. Keďže žiadna hodnota nieje menšia ako δ, aproximovaný vektor $\hat{w}_0$ = [1 0 0 0] nezanedbal žiadnu pravdepodobnosť. Aktívny stav je v tomto prípade iba jeden a to prvý $A_0 = f0g$. Pomocou adaptívnej uniformizácie vypočítame uniformizačnú maticu.

$$Q_0 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \hat{\lambda}_0 = 1, \ P_0^{unif(C)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

Doterajšie výpočty sme už videli pri štandardnej uniformizácii. Ďalší krok potrebný k výpočtu je proces narodenia. Jedná sa o prvý vektor procesu narodenia, preto bude postačovať $w^B_{(0)}(i+1) = w^B_{(0)}(i) * p_{i;i}$. Vektor $w^B_{(k-1)}$ je zatiaľ nulový. Prvý prvok bude mať pravdepodobnosť 1 $w^B_0(0) = 1$, jedná sa o iniciálny prvok a od neho sa bude odvíjať celý nasledujúci výpočet. Rýchlosť prechodu sa rovná $p_{0;0} = 1 - \frac{1}{5}$. Pomocou týchto hodnôt dopočítame proces narodenia.

$$w^B_0 = [\ 1 \quad 0,8 \quad 0,64 \quad 0,512\ ]$$

Hodnota β_0 nám udáva pravdepodobnosť, že sme urobili 0 krokov, vypočítali sme ju skalárnym súčinom medzi vektorom procesu narodenia a vektorom váh $\beta_0 = w^B_0 \cdot \gamma$. Následne pripočítame β_0 k β. Keďže β je stále menšia $\beta < \epsilon$, pokračujeme vo výpočte.

$$\beta_0 = 0.7158, \ \beta = 0.7158$$

Pripočítame pravdepodobnosť procesu narodenia v každom kroku $\hat{\pi}(i) = \hat{\pi}(i) + \hat{w}_0(i) * \beta_0$. Výsledkom je skutočný vektor $\hat{\pi}$.

$$\hat{\pi} = [\ 0,7158 \quad 0 \quad 0 \quad 0\ ]$$

Postupne aplikujeme ten istý postup pokiaľ nebude splnená podmienka $\beta > \epsilon$.

$$w_1 = [\ 0 \quad 1 \quad 0 \quad 0\ ] \ \hat{w}_1 = [\ 0 \quad 1 \quad 0 \quad 0\ ] \ A_1 = f1g,$$

$$Q_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \hat{\lambda}_1 = 4, \ P_1^{unif(C)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

Keďže už máme aj predchádzajúci vektor $\hat{w}_{k-1}$, bude výpočet procesu narodenia trochu komplikovanejší. Najskôr si určíme rýchlosti $p_{0;1} = \frac{1}{5}$ a $p_{1;1} = 1 - \frac{4}{5}$, ktoré budeme potrebovať k výpočtu. Postupným výpočtom pomocou vzorcu $w^B_{i+1}(1) = w^B_i(1) * \frac{1}{5} + w^B_i(1) * \frac{1}{5}$ vypočítame nasledujúci vektor procesu narodenia. Na prvom indexe sa nachádza 0. Znamená že v čase 0 nemôžeme byť v stave 1.

$$w^B_1 = [\ 0 \quad 0,2 \quad 0,2 \quad 0,168\ ],$$

$$\beta_1 = 0.1382, \ \beta = 0.8540, \ \hat{\pi} = [\ 0,7158 \quad 0,1382 \quad 0 \quad 0\ ]$$

Keďže nebola splnená podmienka $\beta < \epsilon$ pokračujeme vo výpočte:

$$w_2 = [\ 0 \quad 0 \quad 0,5 \quad 0,5\ ]\ \hat{w}_2 = [\ 0 \quad 0 \quad 0,5 \quad 0,5\ ]\ A_2 = \{2, 3\},$$

$$Q_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 4 & 1 & 0 \end{pmatrix},\ \hat{\lambda}_2 = 5,\ P_2^{unif(C)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3/5 & 0 & 2/5 & 0 \\ 0 & 4/5 & 1/5 & 0 \end{pmatrix},$$

$$p_{1;2} = \frac{4}{5},\ p_{2;2} = 1 - \frac{5}{5},\ w_2^B = [\ 0 \quad 0 \quad 0,16 \quad 0,16\ ]$$

Pri pripočítaní $\beta_2 = 0,0602$ k β nám vyšla pravdepodobnosť väčšia $\beta = 0,91443$ ako presnosť ϵ, teda môžeme skončiť výpočet. Výsledný vektor pravdepodobnosti bude:

$$\hat{\pi} = [\ 0,7158 \quad 0,1382 \quad 0,0301 \quad 0,0301\ ]$$

Nemuseli sme vykonať ďalší cyklus výpočtov, pretože sme už dosiahli splnenú presnosť. Celková chyba nášho výpočtu je $1 - \sum_{(s2S)} \hat{p}_x(t) = 0,08557$.

Kapitola 6

Implementácia

6.1 Storm

Storm je nástroj, ktorý slúži na analýzu systémov zahŕňajúcich náhodné alebo pravdepodobnostné javy. Či už sa jedná o distribuované algoritmy, bezpečnosť, vstavané systémy alebo už mnoho krát spomínaná biológia. Tento nástroj je open source od roku 2017. Je to nový nástroj, ktorého zámerom je byť flexibilnejší a rýchlejší ako jemu podobné nástroje, napríklad PRISM [15], MRMC [13] a ISACMC [10]. Storm je v súčastnosti state of art, teda dosahuje najvyššiu úroveň zo súčasných nástrojov.

Základnou charakteristikou Stormu je:

- podporovanie rôznych natívnych formátov vstupu: vstupný formát v Prism , generalizované stochastické Petriho siete, dynamické stromy porúch
- podpora Markovových reťazcov, MDPs (Markovove rozhodovacie procesy), Markovov automat, model obsahujúci pravdepodobnostné rozvetvenie, nedeterminizmus a exponenciálne rozložené oneskorenia
- explicitná alebo plne symbolická kontrola modelu ale aj spojenie týchto modelov
- modulárna zostava umožňujúca jednoduché menenie rôznych riešiteľov (angl. solvers) a odlišných balíkov rozhodovacích diagramov
- poskytovanie rozhrania $PythonAPI$, ktoré uľahčuje jednoduché a rýchle vytváranie prototypov iných nástrojov pomocou prostriedkov a algoritmov Stormu
- poskytovanie nasledovných funkcionalít: syntézu protipríkladov, herné abstrakcie MDPs nekonečnými stavmi, efektívne algoritmy pre podmienené pravdepodobnosti a odmeny, dlhodobé priemery na MDPs
- výkon z hľadiska rýchlosti overovania a pamäte v prístroji PRISM benchmark je väčšinou lepší v porovnaní s nástrojom PRISM [6]

Pri pravdepodobnostnej kontrole modelu existujú dva rozhodujúce aspekty a to je efektívnosť z hľadiska času a priestoru. Storm má ako jeden z cieľov poskytovať dobrý priestorovo-časový kompromis pri riešení týchto úloh. Pravdepodobnostná kontrola modelu je náročná úloha pretože väčšina techník vyžaduje, aby bol v pamäti dostupný celý systém a spolieha sa na riešenie obrovských rovníc.

V tejto práci budeme implementovať už vysvetlenú rýchlu adaptívnu uniformizáciu (sekcia 5), ktorú tento nástroj neobsahuje. Pri výpočte budeme využívať Fox Glynnov algoritmus, ktorý je už v Storme implementovaný. Pre úsporu dát využívame riedke matice.

6.1.1 Riedka matica

Matice ktoré obsahujú zväčša 0 sa nazývajú riedke. Sú využívane v informatike ale aj pri numerickej analýze. Maticu môžeme považovať za riedku ak je viac ako polovica prvkov 0. Naopak ak je väčšina prvkov nenulová nazýva sa matica hustá. Zatiaľ čo pri hustej matici si potrebujeme zapamätať všetky hodnoty pri riedkej nám stačia nenulové. Na ukladanie týchto matíc využijeme bezstratovú kompresiu dát. Ak by sme riedku maticu ukladali kompletnú bolo by to značné plytvanie pamäťou počítača.

6.1.1.1 Komprimovaná riedka matica

Pri menších maticiach nieje úspora tak viditeľná. Avšak pri maticiach ktoré obsahujú tisícky stavov nám táto komprimácia ušetrí množstvo miesta a výpočtov.

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 5 & 0 & 0 \\ 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 7 \end{pmatrix}$$

Vektor hodnôt V reprezentuje všetky hodnoty, ktoré nie sú 0.

$$V = [\ 1 \quad 5 \quad 5 \quad 4 \quad 4 \quad 9 \quad 2 \quad 2 \quad 7\ ]$$

Nasledujúci vektor C nám určuje, z ktorého stĺpca hodnota pochádza. Prvá hodnota 1 pochádza zo stĺpca 0, druhá hodnota 5 pochádza z 3. stĺpca.

$$C = [\ 0 \quad 3 \quad 4 \quad 0 \quad 2 \quad 3 \quad 1 \quad 3 \quad 5\ ]$$

Posledný vektor R nám určuje poradie čísla pre všetky prvé čísla v danom riadku. V prvom riadku je prvé číslo 1 a je zároveň aj prvé číslo v V takže dostane hodnotu 0. V druhom riadku je prvé číslo 5 toto číslo je 2. číslo takže dostane hodnotu 1. Pri 3. riadku už však máme prvé číslo 4 ale nieje 3. ale až 4. v poradí vo vektor V takže má hodnotu 3 [18].

$$R = [\ 0 \quad 1 \quad 3 \quad 5 \quad 6 \quad 7\ ]$$

6.2 Rýchla adaptívna uniformizácia

Základný cyklus metódy FAU, ktorý počíta od 0 po R alebo pokiaľ $\beta < \epsilon$ sa skladá z 3 základných častí. Pred hlavným cyklom je však ešte potrebné zistiť λ. Využitím algoritmu Fox Glynn dostaneme body ohraničenia R a L a Poissonovo rozdelenie γ. Deklarujeme a definujeme si všetky potrebné premenné, ktoré budeme využívať. Zvolíme si prah δ a presnosť ϵ a následne môžeme prejsť na hlavnú časť algoritmu.

6.2.1 Proces narodenia

Pri procese narodenia postupujeme od 0 po pravé ohraničenie R. Po dosiahnutí ľavého ohraničenia L pri výpočte vektoru $newBirthProcess$ začneme počítať hodnotu β. Hodnoty $newBirthProcess$ násobíme Poissonovým rozdelením, ktoré vypočítal Fox Glynnov algoritmus.

6.2.2 Tranzientná analýza

Pred výpočtom tranzientnej analýzy si nevytvárame celú maticu $P^{unif(C)}$. Postupne si dopočítavame iba hodnoty, ktoré potrebujeme. Násobíme teda postupne hodnoty vektoru π s vypočítanou hodnotou pomocou rýchlosti λ_k vypočítanej pri filtrovaní aktívnych stavov. Vo vektore $activeStates$ máme uložené indexy aktívnych stavov, takže nemusíme prechádzať celý stavový vektor ale iba aktívne stavy.

Následne si vytvárame vektor indexov stavov, ktoré sa zmenili $checkedActiveStates$. Budeme ho využívať pri zisťovaní aktívnych stavov. Dôvod prečo hneď nezisťujeme, či je daný stav aktívny je ten, že v priebehu propagácie pravdepodobnosti sa môže niekoľkokrát navýšiť pravdepodobnosť, že sa dostaneme do toho stavu. Do jedného stavu sa môžeme dostať z viacerých stavov.

6.2.3 Filtrovanie aktívnych stavov

Pri filtrovaní aktívnych stavov využívame vektor $checkechActiveStates$. Nemusíme prechádzať teda celý vektor π, stačí porovnávať hodnoty, ktoré boli zmenené s δ. Zároveň vytvoríme vektor aktívnych stavov $activeStates$, ktorý využijeme pri tranzientnej analýze.

Ďalej tento cyklus využívame na zistenie λ_k zo stavov, ktoré vyhodnotíme ako aktívne. Túto hodnotu si pripravíme pre ďalší beh cyklu.

Rovnako ako λ_k aj získavanie výsledného stavového vektora π prebieha hneď po prehlásení stavu za aktívny. Teda násobíme hodnoty vektora $\pi(k)$ s $\beta(k)$.

6.3 Štandardná uniformizácia

Pri SU je postup podobný ako pri FAU, hlavný cyklus obsahuje filtrovania aktívnych stavov aj tranzientnú analýzu. Namiesto výpočtu procesu narodenia pri filtrovaní aktívnych stavov násobíme stavový vektor Poissonovým rozdelením γ vypočítaným z Fox Glynnovho algoritmu. Zásadná zmena je v hlavnom cykluse, pretože počíta až do R nemôže skončiť skôr. Ďalšia podstatná zmena je, že pri priebehu tranzientnej analýzy nemusíme dopočítavať hodnoty matice $P^{unif(C)}$, pretože si ju vypočítame ešte pred hlavným cyklom.

Kapitola 7

Experimenty

Pomocou experimentov budeme porovnávať časy výpočtu nástroju Storm a doimplementovanej štandardnej uniformizácie a rýchlej adaptívnej uniformizácie. Experimenty boli vykonávané na počítači s procesorom Intel Core i7 4510 a 8GB RAM na systéme Linux Ubuntu 17.04. Budeme vykonávať sady experimentov na dokázanie efektívnejšieho výpočtu doimplementovaných SU a FAU v protiklade so Stormom, ktorý tieto metódy neobsahuje.

7.1 Modely

Na nasledujúcich modeloch predstavených v tejto sekcii vykonáme sadu experimentov. Všetky modely sú implementované v jazyku PRISM.

7.1.1 Epidemický model

Jednoduchý matematický model epidémie sa často označuje aj ako SIR model. Je vhodný na predikciu ochorení, ktoré sa prenášajú z človeka na človeka ako napríklad osýpky alebo ružienka. Simuluje tiež zotavenie z ochorenia a získanú imunitu na túto chorobu.

- S - náchylné (angl. susceptible)
- I - infikované (angl. infected)
- R - vyliečené (anlg. recovered)

S sú zdravé molekuly, ktoré ešte neboli napadnuté infekciou. Nie je u nich vytvorená imunita a preto môžu byť napadnuté. Infikované molekuly I majú chorobu a môžu ju prenášať na zdravé molekuly. Molekuly ktoré sa už vyliečili z choroby a sú imúnne na túto chorobu sa označujú R.

Na začiatku máme počet molekúl N. Predpokladáme, že väčšina populácie je zdravá a pár molekúl je infikovaných. S rýchlosťou ki môže I nakaziť S. Podobne s rýchlosťou kr sa môže I vyliečiť a zmeniť na R. V momente, keď neexistuje žiadna I molekula nenastáva už žiadna reakcia.

Reakcie:

- $S + I \xrightarrow{ki} 2I$
- $I \xrightarrow{kr} R$

7.1.2 Lotka-Volterr model

Model Lotka-Volterra sa nazýva aj jednoduchým modelom dravec-korisť. Tento model sa môže používať pri simulácii zvierat v prírode. Môže sa jednať o populácie líšok a zajacov v lese, rýb v jazere alebo aj hmyzu.

- D - dravec (angl. predator)
- Y - korisť (angl. prey)

Korisť Y prosperuje v prípade ak neexistuje žiadny dravec D, prípadne ich je málo. S rozmnožením dravcov D sa znižuje populácia Y. V prípade ak je nedostatok Y, tak ubúda aj z populácie D pretože bez Y nemôže prosperovať. Veľkosť modelu ovplyvňujeme pomocou hodnoty N, ktorá vyjadruje maximálny počet jedincov. V prípade ak je Y bez D má prirodzenú rýchlosť nárastu populácie ky. Ďalej potrebujeme rýchlosť úmrtnosti kd populácie D v prípade neprítomnosti Y. Posledným údajom bude ako sa navzájom ovplyvňujú Y a D, teda rýchlosť ko reprodukcie D po skonzumovaní Y.

Reakcie:

- $\xrightarrow{ky}$ Y
- Y + R $\xrightarrow{kd}$ 2R
- R $\xrightarrow{ko}$

Populácia Y rastie s určitou rýchlosťou a zároveň zaniká pri vytvorení nového R. Naopak populácia R umiera a reprodukuje sa iba po skonzumovaní Y.

7.1.3 Model dvojzložkovej signalizačnej cesty

Predstavíme si komplikovanejší model dvojzložkovej signalizačnej cesty. Táto cesta je daná 4 druhmi a 9 cestami. Aby bola analýza uskutočniteľná, obmedzili sme stavový priestor celkových populácii v intervaloch 25 ≤ $H + Hp$ ≤ 35 a 25 ≤ $R + Rp$ ≤ 35. Toto skracovanie má významný vplyv na distribúciu premennej Rp, ktorá reprezentuje vstupný signál [1].

- H - histidín kináza
- R - regulátor odozvy
- Hp - fosforyláciová forma
- Rp - fosforyláciová forma

Model reakčnej signálnej cesty, kde $X \in \{H, Hp, R, Rp\}$ a $Y \in \{H, Rg\}$. Všeobecná reakcia prenosu signálu

- H + S $\xrightarrow{0.1}$ Hp + S
- Hp + R $\xrightarrow{0.1}$ H + Rp
- Rp $\xrightarrow{1.0}$ R

Degradácia/syntéza zložiek

- X $\xrightarrow{0.01}$
- $\xrightarrow{sig(n)}$ Y pre $sig(n) = \frac{0.6}{1+\frac{Y}{30}^{\pi}}$

7.2 Experimenty

V tabuľke stĺpec s názvom *aktívne* označuje priemerný počet aktívnych stavov v stavovom vektore. V stĺpci s označením *cykly* nájdeme počet iterácii hlavného cyklu. Čas sme merali iba v rámci výpočtu algoritmov. Nie je v ňom započítaný čas, počas ktorého Storm vytváral maticu prechodovej rýchlosti, alebo vykonával svoju réžiu potrebnú k behu, keďže sme do týchto častí implementačne nezasahovali. Doba trvania tejto časti bude pri všetkých metódach rovnaká. Hodnota prahu sa bude často meniť, ale hodnota presnosti ϵ bude pri pokusoch rovná $1e-7$, ak nebude uvedené inak. Pri pokusoch je v tabuľkách aj texte pre lepší formát uvádzaný ϵ ako $\epsilon = 1 - \epsilon$.

7.2.1 Experimenty s epidemickým modelom

Začneme so sadou experimentov s modelom SIR. Zvolíme hodnotu maximálnej populácie 1450 a určíme množstvo infikovaných stavov 5. Výpočet tohto modelu trval Stormu 165.19s, čo je 226 krát pomalšie oproti metóde FAU, ktorej to priemerne trvalo 0.743s. Táto hodnota je priemerom časov z tabuľky 7.1, stĺpcu FAU čas. Počet stavov pri experimente je 1053411, avšak metóda FAU počíta priemerne len s 664 stavmi. Množstvo stavov s ktorými počíta, je teda 1586 krát menšie.

Pri porovnaní Stormu s SU zistíme, že čas sa zrýchlil 150 násobne a stavový priestor sa zmenšil 1848 krát. V ideálnom prípade by sa však čas zmenšil toľko krát koľko sa zmenšil stavový priestor. Ak by sme počítali s o 100 krát menej stavmi aj čas by teda mal byť 100 krát menší. Avšak čas výpočtu zahrňuje aj zisťovanie aktívnych stavov, tvorenie uniformizačnej matice či vytváranie vektora zmenených stavov.

	SU				FAU			
Prah	Čas	Chyba	Aktívne	Cykly	Čas	Chyba	Aktívne	Cykly
1e-9	0.988	7.6e-06	342	1564	0.74	9.7e-08	569	86
1e-12	1.13	3.1e-09	590	1077	0.74	8.0e-08	684	90
1e-15	1.177	1.0e-10	779	1077	0.75	8.5e-08	739	91

Tabuľka 7.1: Experiment s približne miliónom stavov

Zrýchlenie FAU oproti SU je len mierne. Pri analýze časovej náročnosti výpočtu som zistila, že počiatočná inicializácia trvá približne 90% času. Inicializácia je takmer rovnaká v obidvoch prípadoch. Z času výpočtu 0,74s trvá inicializácia až 0,70s. V jej časti sa nachádza Fox Glynnov algoritmus, deklarácie, zisťovanie uniformizačnej rýchlosti. Pri SU obsahuje aj výpočet uniformizačnej matice, ktorá túto časť spomalí. Zaujímavé je, že i keď FAU počíta priemerne s viac stavmi ako SU, urobí omnoho menej cyklov. Dynamika tohto experimentu je malá, keďže je málo infikovaných buniek.

Ďalej nás bude zaujímať, či zrýchlenie bude stále rovnaké aj v prípade väčšieho množstva aktívnych stavov pri väčšom modeli. Nastavili sme teda veľkosť populácie na 2150 a z nich sme infikovali 700 molekúl. Veľkosť stavového priestoru je tentokrát 2069126x2069126. Storm príklad počítal 583.92s.

V tomto prípade už inicializácia nezaberala väčšinu času výpočtu, preto môžeme v tabuľke 7.2 vidieť už výraznejšie zrýchlenie FAU oproti SU. Na druhú stranu oproti Stormu sa zrýchlenie zmenšilo na 66 krát. Dôvod tohto poklesu je, väčšia dynamika modelu ktorá

	SU				FAU			
Prah	Čas	Chyba	Aktívne	Cykly	Čas	Chyba	Aktívne	Cykly
1e-9	8.09	1.2e-04	10032	1532	3.96	5.2e-05	5498	1014
1e-12	11.85	1.4e-07	16121	1532	9.58	1.6e-07	9043	1033
1e-15	15.86	1.6e-10	22084	1532	12.99	9.1e-08	12709	1049

Tabuľka 7.2: Experiment s navýšením aktívnych stavov

mala za príčinu zvýšenie množstva aktívnych stavov. Teda stavový priestor sa zmenšil len 227 krát.

V prípade ak sa pozrieme lepšie na chybu vidíme že v 1. výpočte je chyba FAU menšia avšak v 3. je väčšia. Tento jav je spôsobený hodnotou presnosti $\lambda = 1e - 7$. FAU po dosiahnutí presnosti 1-ϵ ukončí výpočet, zatiaľ čo SU počíta ďalej.

V experimente číslo 3 sme zisťovali ako ovplyvňuje hodnota presnosti výpočet FAU. Prah sme nastavili na pevnú hodnotu 1e-12 a modifikovali sme teda hodnotu epsilon. Tentokrát sme navýšili množstvo infikovaných buniek na 500 a populácia bola veľkosti 1450. Počítali sme s 1008276 stavmi.

Epsilon	Čas	Chyba	Aktívne	Cykly
1e-3	2.41	9.5e-04	6328	569
1e-5	2.86	1.0e-05	6721	598
1e-7	2.87	1.2e-07	7049	620
1e-9	2.85	3.1e-08	7324	642
1e-11	2.95	3.0e-08	7572	660

Tabuľka 7.3: Experiment s prahom

Z tabuľky 7.3 je možné vidieť, ako sa postupne zvyšuje množstvo cyklov aj aktívnych stavov pre dosiahnutie potrebnej presnosti. Pri každom zmenení presnosti mierne klesá rozdiel medzi cyklami. Časy výpočtu sa líšia len mierne.

7.2.2 Experimenty s Lotka-Volterr modelom

V prvom experimente s týmto modelom sme nastavili maximálnu veľkosť populácie na 1000. Snažili sme sa nastaviť podobné množstvo stavov ako mal pri prvom experiment model SIR. Model experimentu má 1002001 stavov. Zaujímalo nás ako sa rôzne modely chovajú pri rovnakom počte stavov. Stormu výpočet trval približne hodinu a pol.

	SU				FAU			
Prah	Čas	Chyba	Aktívne	Cykly	Čas	Chyba	Aktívne	Cykly
1e-9	108.13	2.0e-03	6233	35156	23.4	3.9e-4	9487	2905
1e-12	194.9	2.7e-06	11355	35156	37.6	8.0e-08	15615	3220
1e-15	2955.3	3.2e-09	16925	35156	54.78	9.7e-08	21970	3509

Tabuľka 7.4: Experiment s približne miliónom stavov

V porovnaní s prvým experimentom SIR, ktorý obsahoval tiež približne 1000000 stavov je výrazne pomalší. Dokonca je pomalší aj ako experiment číslo 2, ktorý prebiehal s dvojnásobným počtom stavov. I keď bolo množstvo aktívnych stavov v experimente 2 (hodnoty v tabuľke 7.2) pri druhom príklade podobné ako v experiment 4 (hodnoty v tabuľke 7.4) pri prvom príklade, počet cyklov bol výrazne väčší. Z toho môžeme konštatovať, že pri FAU čas výpočtu nezáleží len od veľkosti daného modelu a počtu aktívnych stavov. Pri ďalšom experimente sme navýšili počet stavov na 2253001. V tomto experimente číslo 5 sa najviac prejavila výpočtová sila metódy FAU. Zvládla do minúty vypočítať príklad, ktorý Stormu trval takmer 7 hodín. Maximálna veľkosť populácie bola 1500.

	SU				FAU			
Prah	Čas	Chyba	Aktívne	Cykly	Čas	Chyba	Aktívne	Cykly
1e-9	127.79	2.6e-03	3612	71768	36.17	3.9e-4	7779	3337
1e-12	267.2	3.7e-06	6824	71768	47.4	6.0-07	12683	3590
1e-15	367.97	1.0e-10	10363	71768	60.27	8.7e-08	17620	3824

Tabuľka 7.5: Experiment s navýšeným množstvom stavov

Rovnako ak sa aj pozrieme do tabuľky 7.5 vidíme mnohonásobné zrýchlenie FAU v porovnaní s SU. Tento príklad môžme brať za takmer ideálny, keďže SU zredukovala stavový priestor 108 krát a čas 97 krát. Vo všeobecnosti jednoduché modely ako epidemický alebo Lotka-Volterr ukazujú pozitívne stránky SU a FAU, pretože nemajú veľkú dynamiku. Sú však aj modely, ktoré sú komplikovanejšie a majú väčšiu dynamiku, pre ktoré to neplatí.

Pri tomto experimente sme nemenili hodnotu prahu, tá bola 1e-11. Modifikovali sme množstvo stavov, nad ktorými prebiehal výpočet.

	SU				FAU			
Stavy	Čas	Chyba	Aktívne	Cykly	Čas	Chyba	Aktívne	Cykly
106276	121.88	1.3e-05	41847	5940	131.85	1.3e-05	43721	5054
303601	164.84	2.0e-05	25183	13102	57.57	7.7e-06	26139	3590
609961	161.48	2.2e-05	14154	23086	35.41	5.6e-06	16815	3141
904401	166.69	2.4e-05	10360	32197	31.92	5.1e-06	14065	3108
1104601	196.47	2.6e-05	8946	38245	33.97	4.9e-06	13127	3138

Tabuľka 7.6: Experiment s meniacou sa veľkosťou modelu

Pri tomto experimente sme objavili zaujímavý jav. Pri malom množstve maximálnej populácie bolo mnoho stavov aktívnych a i keď FAU spravila takmer o 1000 cyklov menej, výpočet jej trval dlhšie ako SU. Dokonca tieto algoritmy spravili rovnakú chybu. V tomto prípade by SU bola o čosi rýchlejšia ako FAU. Avšak je vidieť, že zo zvyšovaním množstva stavov, opäť FAU preukazuje, že je rýchlejšia a presnejšia.

7.2.3 Experiment s modelom dvojzložkovej signalizačnej cesty

V tomto experimente sme chceli zistiť aké bude zrýchlenie FAU pri komplikovanom modely. Pretože doteraz v žiadnom modely nebola vysoká dynamika. Stormu trval výpočet 3372.156s pracoval s 600625 stavmi. Pri tomto experimente sa ukázala slabá stránka me-

	SU				FAU			
Prah	Čas	Chyba	Aktívne	Cykly	Čas	Chyba	Aktívne	Cykly
1e-9	1204.76	8.6e-02	45835	25754	743.04	0.4e-2	56129	11228
1e-12	3719.82	1.7e-04	144567	25754	2385.3	9.2e-05	157654	12652
1e-15	6364.5	2.2e-07	262983	25754	4565.19	8.7e-08	276386	13918

Tabuľka 7.7: Experiment na zložitom modely

tódy FAU. Ak budeme od FAU chcieť väčšiu presnosť bude musieť počítať až s takmer polovičným stavovým priestorom spraviť skoro 14000 cyklov. I keď Storm spraví 262983 cyklov s 600625 stavmi je rýchlejší. FAU totiž počíta navyše proces narodenia, filtruje aktívne stavy, dopočítava uniformizačnú maticu či počíta maximálnu rýchlosť. Z tohto dôvodu pri pohľade na celkový čas nie je schopná predbehnúť Storm. Na druhú stranu pri nízkej presnosti je síce FAU rýchlejšia ale chyba je pomerne veľká, takže výsledky by nemuseli byť použiteľné. Avšak FAU je rýchlejšia od SU, keďže stále môže ukončiť výpočet skôr.

7.2.4 Zhrnutie experimentov

Z experimentov sa dá vyvodiť, že čím je aktivita modelu menšia tým je FAU rýchlejšia. Pri malej aktivite vie zredukovať čas výpočtu dokonca 500 násobne. Na druhú stranu v prípade veľkého počtu aktívnych stavov stráca svoje výhody. Ak je model zložitý môže byť pomalšia ako Storm. Vo väčšine prípadov však FAU urýchľuje výpočty v porovnaní s SU. Existuje množstvo modelov pre ktoré je metóda FAU rýchla aj pri malej chybe. Sú však aj modely pre ktoré to neplatí a pri očakávanej malej chybe je FAU pomalšia. Zrýchlenie je vždy závislé od dynamiky daného modelu.

Kapitola 8

Záver

V práci som vysvetlila, ako modelovať biochemické systémy pomocou stochastických modelov. Predstavila som tieto modeli, teda Markovove reťazce v diskrétnom a potom aj v spojitom čase. Následne som podrobne objasnila aproximačné techniky, štandardnú a adaptívnu uniformizáciu. Načrtla som Fox Glynnov algoritmus, ktorý využívam pri výpočte Poissonových pravdepodobností. Po vysvetlení základnej problematiky som objasnila fungovanie rýchlej adaptívnej uniformizácie. Teoretické časti sú doplnené praktickými príkladmi pre lepšie pochopenie danej problematiky.

Rýchlu adaptívnu uniformizáciu som implementovala do nástroju Storm, ktorý sa v súčastnosti radí medzi najvýkonnejšie nástroje orientované na pravdepodobnostné overovanie modelov. Tento nástroj využíva zložité konštrukcie jazyka C++, preto bolo náročné rozširovať jeho funkcionalitu. Okrem rýchlej adaptívnej uniformizácie som pre demonštráciu rozdielnych vlastností implementovala aj štandardnú uniformizáciu.

Pomocou experimentov som si overila výhody rýchlej adaptívnej uniformizácie. Tieto experimenty som vykonala na epidemickom, na modeli Lotka-Volterr a nad modelom dvojzložkovej signalizačnej cesty. Pomocou rôznych hodnôt presnosti výpočtov, veľkosti a parametrov modelov som zisťovala dopad na rýchlosť a chybu výpočtu. Rýchla adaptívna uniformizácia bola vo väčšine prípadov mnohonásobne rýchlejšia ako Storm. Na druhú stanu jej efektivita značne klesla pri náročnom dynamickom modely. Počas experimentov som zistila aj značné zrýchlenie oproti štandardnej uniformizácii.

Literatúra

[1] Abate, A.; Brim, L.; Češka, M.; aj. Adaptive aggregation of Markov chains: Quantitative analysis of chemical reaction networks. 2015.

[2] Brim, L.; Češka, M.; Šafránek, D. : Model checking of biological systems. In *Formal Methods for Dynamical Systems*, Springer, 2013, s 63–112.

[3] Burak, M. R. : Computing discrete poisson probabilities for uniformization algorithm. *Studia Informatica*, roč. 38, č. 1B, 2017: s 77–88.

[4] Cassandras, C. G.; Lafortune, S. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[5] Ciocchetta, F.; Gilmore, S.; Guerriero, M. L.; aj. : Integrated simulation and model-checking for the analysis of biochemical systems. *Electronic Notes in Theoretical Computer Science*, roč. 232, 2009: s 17–38.

[6] Dehnert, C.; Junges, S.; Katoen, J.-P.; aj. A storm is coming: A modern probabilistic model checker. 2017.

[7] Fox, B. L.; Glynn, P. W. : Computing poisson probabilities. *Communications of the ACM*, roč. 31, č. 4, 1988: s 440–445.

[8] Garfinkel, D. : Construction of biochemical computer models. *FEBS letters*, roč. 2, č. S1, 1969.

[9] Gupta, A.; Henzinger, T. A. *Computational Methods in Systems Biology 11th International Conference, CMSB 2013, Klosterneuburg, Austria, September 22-24, 2013. Proceedings*. Springer Berlin Heidelberg, 2013.

[10] Hahn, E. M.; Li, Y.; Schewe, S.; aj. iscas M c: a web-based probabilistic model checker. 2014.

[11] Henzinger, T. A.; Mikeev, L.; Mateescu, M.; aj. Hybrid numerical solution of the chemical master equation. 2010.

[12] Higham, D. J. : Modeling and simulating chemical reactions. *SIAM review*, roč. 50, č. 2, 2008: s 347–368.

[13] Katoen, J.-P.; Zapreev, I. S.; Hahn, E. M.; aj. : The ins and outs of the probabilistic model checker MRMC. *Performance evaluation*, roč. 68, č. 2, 2011: s 90–104.

[14] Kwiatkowska, M.; Norman, G.; Parker, D. Stochastic model checking. 2007.

[15] Kwiatkowska, M.; Norman, G.; Parker, D. PRISM 4.0: Verification of probabilistic real-time systems. 2011.

[16] Mateescu, M.; Wolf, V.; Didier, F.; aj. : Fast adaptive uniformisation of the chemical master equation. *IET systems biology*, roč. 4, č. 6, 2010: s 441–452.

[17] Mateescu, M.-E.-C. : Propagation models for biochemical reaction networks. 2011.

[18] Saad, Y. : SPARSKIT: A basic tool kit for sparse matrix computations. 1990.

[19] Van Moorsel, A. P.; Sanders, W. H. : Adaptive uniformization. *Stochastic Models*, roč. 10, č. 3, 1994: s 619–647.