Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie realizácií rovnomerného rozdelenia pravdepodobnosti na mnohorozmerných polyédroch

Bakalárska práca

2018
Slavomír Hanzely

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie realizácií rovnomerného rozdelenia pravdepodobnosti na mnohorozmerných polyédroch

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Školiteľ:	doc. Mgr. Radoslav Harman, PhD.

Bratislava, 2018
Slavomír Hanzely

55658269

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Slavomír Hanzely
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Generovanie realizácií rovnomerného rozdelenia pravdepodobnosti na mnohorozmerných polyédroch
Random sampling from the uniform distribution on multidimensional polyhedra

Anotácia: V Monte-Carlo metódach výpočtu pravdepodobností a v znáhodnených optimalizačných metódach je často potrebné generovať realizácie z rovnomerného rozdelenia na mnohorozmerných polyédroch. Tieto polyédre môžu byť zadané buď systémom konečného počtu lineárnych nerovníc (takzvaná H-reprezentácia), alebo ako konvexný obal konečnej množiny bodov (takzvaná V-reprezentácia). V prípade oboch typov reprezentácií je rovnomerné generovanie vo vnútri všeobecného polyédra netriviálna úloha, kombinujúca techniky a poznatky z matematiky, štatistiky a informatiky.

Cieľ: Cieľom bakalárskej práce je: Po prvé vypracovať prehľad existujúcich prístupov generovania realizácií z rovnomerného rozdelenia na polyédroch (priame generovanie pre špeciálne polyédre, zamietacie algoritmy, MCMC algoritmy a iné); po druhé vypracovať a programovo implementovať vlastnú metódu založenú na elipsoide najmenšieho objemu obsahujúceho zadaný polyéder.

Vedúci:	doc. Mgr. Radoslav Harman, PhD.
Katedra:	FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry:	prof. RNDr. Daniel Ševčovič, DrSc.
Dátum zadania:	14.10.2018

Dátum schválenia: 24.10.2018

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

študent

vedúci práce

Poďakovanie:

Abstrakt

Kľúčové slová:

Abstract

Keywords:

Obsah

Úvod

V rámci tejto práce sa budeme zaoberať metódami na rovnomerné generovanie bodov vo veľarozmernom polyédre (konvexnom mnohostene). Našou úlohou vytvoriť generátor, ktorý bude čo najrýchlejšie generovať body vnútri polyédru rovnomerne náhodne, tj. pravdepodobnosť, že dostaneme bod vnútri ľubovoľnej oblasti polyédra je lineárne závislá iba od objemu danej časti.

Vo všeobecnosti možno polyéder reprezentovať viacerými spôsobmi, napríklad ako konvexný obal bodov (V-reprezentácia) alebo ako sústavu lineárnych nerovníc (H reprezentáca). Obidve spomenuté reprezentácie možno v prípade potreby previesť na tú druhú. Prevod medzi nimi síce nie je lacný, no daný výpočet je nutné spraviť len raz pred začatím generovania. Pre účely tejto práce budeme pracovať s polyédrom reprezentovaným sústavou lineárnych nerovníc, riešení systému $Ax \leq b$ ($x \in X$ ak $Ax \leq b$).

Rovnomerné generovanie bodu v polyédre je problém s prirodzeným uplatnením v praxi. Mnoho algoritmov, napríklad z triedy Monte Carlo alebo z triedy znáhodnených optimalizačných metódach, je závislých na rovnomernom generovaní bodov splňujúcich určité požiadavky. Generovanie bodov v polyédre predstavuje generovanie bodov, ktoré spĺňajú sústavu lineárnych obmedzení (viď H-reprezentácia polyédru).

Ako základ náhody bude náš generátor bodu v polyédre používať rovnomerný generátor čísel $[0, 1]$. Pomocou generátoru na $[0, 1]$ možno triviálne generovať bod na $[0, k]$ (prenásobením konštantou k), tiež možno generovať bod na $[a, b]$ (vygenerovaním bodu na $[0, -a + b]$ a pripočítaním konštanty a, alebo bod na $[0, 1]^n$ (postupným vygenerovaním súradníc).

Cieľom tejto práce je jednak poskytnúť prehľad známych metód, ktoré je možné použiť na rovnomerné generovanie v polyédroch a taktiež implementovať čo najefektívnejší generátor.

V prvej kapitole sa budeme zaoberať známymi metódami, ktoré možno použiť na generovanie na polyédroch. Medzi ne patria Metropolis-Hastings metódy (z triedy Markov Chain Monte Carlo), ktoré sa snažia simulovať komplexné distribúcie priamim výberom. To možno použiť aj v našom prípade, keď je cielená distribúcia uniformná. Okrem toho sa budeme zaoberať aj zamietacími metódami, ktoré namiesto generovania bodov priamo v polyédri vygeneruju bod na jednoduchšej nadmnožine polyédra rovnomerne náhodne. Po vygenerovaní bodu overia, či leží v polyédre. Ak nie, tak generujú znovu.

Druhá kapitola je venovaná problému optimálneho riadenia (optimal design problem), pomocou ktorého predstavíme algoritmus Randomized Exchange Algorithm. Daný algoritmus vieme použiť aj na nájdnenie elipsoidu s minimálnym objemom obaľujúci zadaný polyéder. Tento elipsoid možno jednak priamo použiť ako nadmnožinu pri zamietacej metóde, no taktiež možno jeho hlavné osi využiť na zistenie natočenia polyédra a obalenie polyédra kvádrom s malým objemom.

Tretia kapitola bude obsahovať porovnania algoritmov spomenutých v prvej kapitole a zdrojový kód čo najrýchlejšieho algoritmu na generovanie bodov vnútri polyédru.

Táto kapitola sa bude meniť

Kapitola 1

Metódy generovania vnútri polyédru

V tejto kapitole sa budeme zaoberať známymi metódami na generovanie z určitej distribúcie, ktorá je v našom prípade rovnomerná vnútri polyédra a nulová mimo polyédra. V prvej sekcii sa budeme zaoberať triedou Metropolis-Hastings algoritmov, v druhej kapitole sa budeme zaoberať zamietacími metódami.

1.1 Metropolis-Hastings metódy

V tejto sekcii si predstavíme triedu Metropolis-Hastings algoritmov na generovanie bodov z ľubovoľnej distribúcie. Na postupnosť bodov generovnaných algoritmami z triedy Metropolis-Hastings sa dá pozerať ako na postupnosť stavov markovovských reťazcoch (Markov Chain Monte Carlo). Daným spôsobom je ich možné analyzovať a dokázať, že generujú body podľa žiadanej distribúcie (kvôli časovému obmedzeniu práce je táto časť vynechaná).

Najprv sa pozrieme na všeobecný Metropolis-Hastings algoritmus, následne sa pozrieme na jeho konkrétne realizácie.

1.1.1 Všeobecný Metropolis-Hastings algoritmus

Majme cieľovú hustotu Q z ktorej chceme generovať, v prípade rovnomerného generovania vnútri polyédru je rovnomerná v polyédri a nulová mimo neho.

Metropolis-Hastings algoritmus je vždy v stave $x^{(i)}$ reprezentovanom bodom v priestore, stav určuje hustotu $Q(x^{(i)})$ závislú na $x^{(i)}$. Algoritmus vygeneruje ďalší potenciálny stav y podľa hustoty $Q(x^{(i)})$. Ďalší stav algoritmu $x^{(i+1)}$ bude y s pravdepodobnosťou $\alpha(x^{(i)}, y)$, inak to bude $x^{(i)}$.

Algorithm 1 Všeobecný Metropolis-Hastings algoritmus [2]

inicializuj $x^{(0)}$
for $i = 0, 1, \ldots, N$ **do**
    Vygeneruj bod y z $Q(x^{(i)})$
    Vygeneruj u z $U(0,1)$.
    if $u \leq \alpha(y|x^{(i)})$ **then**
        Nastav $x^{(i+1)} = y$
    else
        Nastav $x^{(i+1)} = x^{(i)}$
Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Môžeme si všimnúť, že v Metropolis-Hastings algoritme je bod $x^{(i)}$ závislý od predchádzajúceho bodu $x^{(i-1)}$. Podľa [2] je možné dokázať, že napriek závislosti po sebe idúcich bodov pre dostatočne veľké N budú body $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ z hustoty Q (potrebná veľkosť N sa nazýva burn-in period).

V ďalších sekciách si ukážeme niekoľko konkrétnych realizácii Metropolis-Hastings algoritmu. Každá z tých metód obsahuje určité predpoklady na distribúciu, z ktorej chceme generovať, no dá použiť aj na rovnomerné generovanie bodov v polyédri.

1.1.2 Hit-and-Run generátor

Ako jedna z možností na realizáciu Metropolis-Hastings algoritmu prichádza do úvahy Hit-and-Run generátor. Algoritmus je analogický s algoritmom Metropolis-Hasting, pričom hustota $Q(x^{(i)})$ je určená priamkou s náhodným smerom cez bod $x^{(i)}$.

Hit-and-Run generátor funguje nasledovne:

Algorithm 2 Hit-and-Run generátor [1]

Inicializuj $x^{(0)}$
for $i = 0, \ldots, N-1$ **do**
    Vygeneruj smer d_i z distribúcie D na povrchu sféry
    Nájdi množinu $S_i(d_i, x^{(i)}) = \{\lambda \in \mathbb{R}; x^{(i)} + \lambda d_i \in S\}$
    Vygeneruj $\lambda_i \in S_i$ podľa hustoty $Q_i(\lambda|d_i, x^{(i)})$
    Zvoľ $y = x^{(i)} + \lambda_i d_i$
    Vygeneruj u z $U(0,1)$.
    if $u \leq \alpha(y|x^{(i)})$ **then**
        Nastav $x^{(i+1)} = y$
    else
        Nastav $x^{(i+1)} = x^{(i)}$
Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Použiteľnosť Hit-and-Run generátora závisí od toho, ako rýchlo vieme generovať smery d_i z distribúcie D. Ak by distribúcia D bola príliš zložitá a nevedeli by sme generovať z nej rýchlo, celý algoritmus by bol pomalý. Našťastie v prípade polyédrov tento problém nenastane, možno vhodne zvoliť distribúciu D (viď kapitola 3, implementácia Hit-and-Run algoritmu).

1.1.3 Gibbsov generátor

V tejto podsekcii sa budeme zaoberať Gibbsovým generátorom, metódou generovania z triedy MCMC vhodnou na generovanie vo viacrozmernom priestore. Na Gibbsov generátor sa možno dívať ako na špeciálny prípad Metropolis-Hastings algoritmu.

Našou úlohou je generovať z n-rozmernej distribúcie Q, pričom z Q nevieme generovať priamo. Predpokladajme, že nevieme použiť priamo Hit-and-Run generátor, lebo $Q(x^{(i)}) = Q(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$ je príliš zložitá na generovanie. Taktiež predpokladajme, že ak $Q(x^{(i)})$ obmedzíme na jeden rozmer, tak v ňom vieme generovať rýchlo, tj. možno generovať rýchlo z $Q(x_j^{(i)}|x_1^{(i+1)}, x_2^{(i+1)} \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, x_{j+2}^{(i)} \ldots, x_n^{(i)})$.

Gibbsov generátor funguje nasledovne:

Algorithm 3 Gibbsov generátor [5]

1: inicializu $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$
2: **for** $i = 1, \ldots, N$ **do**
3:     **for** $j = 0, 1, \ldots, n$ **do**
4:         $x_j^{(i)} \sim Q(x_j|x_1^{(i+1)}, x_2^{(i+1)} \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, x_{j+2}^{(i)} \ldots, x_n^{(i)})$
5:     $x^{(i+1)} = (x_1^{(i+1)}, x_2^{(i+1)} \ldots, x_n^{(i+1)})$
6: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Ako špeciálny prípad Metropolis-Hastings algoritmu má Gibbsov generátor podobné vlastnosti ako Metropolis-Hastings algoritmus. Jeho hlavnou výhodou je, že je jednoduchý a že neobsahuje žiadne parametre.

Gibbsov generátor možno použiť aj pri rovnomernom generovaní v polyédroch, vďaka linearite nerovníc (pri H-reprezentácii polyédra) sa môžeme ľahko obmedziť na jeden rozmer.

1.1.4 Slice sampling

Slice sampling je metóda generovania z triedy Metropolis-Hastings algoritmov, ktorá je vhodná na generovanie pre distribúcie s určitou štruktúrou. Majme n rozmernú hustotu $Q(x) : \mathbb{R}^n \to [0, \infty)$, pričom $\pi(x) = \Pi_{j=0}^{K} f_j(x)$ pre nejaké $f_j : \mathbb{R}^n \to [0, \infty)$.

Slice sampler bude pracovať nasledovne. Začne s bodom $x^{(0)}$, na vygenerovanie bodu $x^{(i)}$ z $x^{(i-1)}$ najprv vygeneruje nezávislé náhodné premenné $y_{i,j}$ v závislosti od $f_j(x^{(i-1)})$. Následne pomocou $y_{i,j}$ vygeneruje bod $x^{(i)}$.

Algorithm 4 Slice sampling generátor [6]

1: inicializuj $x^{(0)}$
2: **for** $i = 1, \ldots, N$ **do**
3: vygeneruj nezávislé náhodné premenné $y_{i,1}, y_{i,2}, \ldots, y_{i,K}$, kde $y_{i,j} \sim U(0, f_j(x^{(i-1)}))$
4: vygeneruj $x^{(i)}$ z distribúcie $f_0(.)\mathbf{1}_{L(y_i)}$, kde $L(y) = \{z \in \mathbb{R}^n; f_j(z) \geq y_{i,j}, i = 1, 2, \ldots, K\}$
5: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Daný algoritmus je závislý jedine od faktorizácie distribúcie $Q(x)$ na $\Pi_{j=0}^{K} f_j(x)$. Ak možno distribúciu Q faktorizovať na niekoľko jednoduchých distribúcii, tak je daná metóda použiteľná v praxi. V prípade rovnomerného generovania na polyédroch danú metódu možno použiť (viď implementácia Slice sampling generátora).

1.2 Zamietacie metódy

Zamietacie metódy nám poskytujú jeden zo spôsobov rovnomerného generovania bodov na určitej množine. Myšlienka za nimi je nasledovná: Označme si X množinu, na ktorej chceme rovnomerne náhodne generovať prvky. Predpokladajme, že nevieme priamo rovnomerne generovať body na X, no vieme rovnomerne generovať na množine S, $X \subset S$.

Náš generátor G_S bude pracovať nasledovne:

Algorithm 5 Zamietacia metóda

1: **for** $i = 0, \ldots, N-1$ **do**
2: **repeat**Vygeneruj bod $x^{(i)} \in S$ rovnomerne náhodne
3: **until** $x^{(i)} \in X$
4: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$

Generátor G_S vygeneruje bod $x \in S$, ak je ten bod aj z X, tak ho vráti ako výstup, inak vygeneruje nový bod $x \in S$. Všimnime si, že generátor G_S je závislý iba od S, generuje na X rovnomerne náhodne.

Očakávaná rýchlosť generovania závisí od toho, koľkokrát G_S vygeneruje bod mimo X. Z rovnomernosti G_S je tá pravdepodobnosť rovná $\frac{|S-X|}{|S|} = 1 - \frac{|X|}{|S|}$. Označme si p_k pravdepodobnosť, že G_S vygeneruje bod z X na k-ty pokus, t.j. najprv $k-1$ krát vygeneruje bod mimo X a potom vygeruje bod z X. Platí $p_k = (1-\frac{|X|}{|S|})^{k-1}\frac{|X|}{|S|}$. Očakávý počet generovaní G_S je $E(G_S) = \sum_0^\infty kp_k = \frac{|X|}{|S|}\sum_0^\infty k(1-\frac{|X|}{|S|})^{k-1} = \frac{|X|}{|S|}\frac{1}{((1-\frac{|X|}{|S|})-1)^2} = \frac{|S|}{|X|}$.

Táto metóda generovania je vhodná, ak je $\frac{|S|}{|X|}$ dostatočne malé, t.j. ak je obal S relatíve malý oproti polyédru X. Ak je $\frac{|S|}{|X|} \sim \infty$, tak je táto metóda nepoužiteľná.

1.2.1 Použitie na generovanie bodu vnútri polyédru

Zamyslime sa nad tým, ako by sme vedeli použiť túto metódu na generovanie bodu vnútri polyédru. Ako množinu možných S, $X \subset S$ môžeme použiť najmenší kváder so stranami rovnobežnými s osami. Vypočítať súradnece kvádra je ľahké, stačí nám to spraviť raz pred (začatím generovania) pomocou lineárneho programovanie.

Žiaľ, pre takúto množinu S môže byť podiel $\frac{|S|}{|X|}$ byť ľubovoľne veľký. Ako príklad na takú množinu X uveďme kváder s obsahom k pozdĺž diagonály kocky $[0,1]^n$, dotýkajúci sa každej steny kocky $[0,1]^n$. Zrejme najmenšia množina S (kváder so stranami rovnobežnými s osami) obaľujúca X je kocka $[0,1]^n$, ktorá má obsah 1. Platí $\frac{|S|}{|X|} = \frac{1}{k}$. Keďže vieme nájsť kváder taký, že sa dotýka stien kocky $[0,1]^n$ a k je ľubovoľne malé, tak očakávaná dĺžka generovania touto metódou (pre danú množinu S) je ľubovovoľne veľká.

Ako ďalšia možná množina S prichádza do úvady elipsoid obaľujúci polyéder. Keďže chceme, aby bol podiel $\frac{|S|}{|X|}$ čo najmenší, budeme skúmať elipsoid s najmenším obsahom obaľujúci polyéder - Minumum Volume Enclosing Elipsoid (ďalej MVEE). Nájsť daný elipsoid a generovať body v ňom vieme pomocou REX algoritmu, ktorému je venovaná ďalšia kapitola.

Môžeme si všimnúť, že MVEE elipsoid obsahuje veľa informácie o tom, ako vyzerá polyéder. Keďže nedegenerovaný elipsoid je jednotková guľa zobrazená regulárnou lineárnou transformáciou, vieme pomocou inverznej transformácie zobraziť elipsoid na jednotkovú guľu. Dané zobrazenie možno vypočítať pomocou osí MVEE elipsoidu.

Keďže najjednoduchšia množina S, v ktorej vieme generovať je kváder, v 3. kapitole sa pozrieme na prípad, keď za S zvolíme kváder, ktorého osi budú zhodné s osami MVEE elipsoidu. Na daný kváder sa dá pozerať ako na kváder s najmenším objemom obaľujúcim MVEE elipsoid. Tiež pre neho platí, že je obrazom kocky $[0,1]^n$ v zobrazení, ktoré zobrazí jednotkovú guľu na MVEE elipsoid. Okrem iného, daný kváder je kváder obaľujúci MVEE elipsoid s najmenším objemom. Pre daný kváder možno spraviť odhad veľkosti $\frac{|S|}{|X|}$ (viď príloha).

Kapitola 2

Metódy na riešenie problému optimálneho riadenia

Cieľom tejto kapitoly je predstaviť Randomized exchange algoritmus [3] (ďalej REX) ako metódu na riešenie problému optimálneho riadenia (optimal design problem). Vďaka ekvivalencii problému optimálneho riadenia a minimum volume enclosing elipsoidu (MVEE) [3] možno REX využiť aj na riešenie MVEE problému. Vzhľadom na dôležitosť MVEE elipsoidu pre túto prácu a kvôli lepšiemu pochopeniu REX algoritmu sa najprv pozrieme na jednoduchšie metódy riešenia problému optimálneho riadenia.

2.1 Základné metódy na riešenie problému optimálneho riadenia

Najprv si predstavíme metódu Subspace Ascend Method (ďalej SAM) ako všeobecnú iteratívnu metódu na riešenie problému optimálneho riadenia a Vertex Exchange Method (VEM) ako jej konkrétnu realizáciu. Následne sa pozrieme na REX algoritmus ako na špeciálny prípad SAM, ktorý kombinuje VEM metódu s pažravým prístupom.

Označme návrh na X ako n-rozmerný vektor w s nezápornými prvkami so súčtom 1. Komponent w_x vektoru w predstavuje počet pokusov v bode $x \in X$ **TODO vysvetlit**. Označme nosnú množinu návrhu w ako $supp(w) = \{x \in X | w_x > 0\}$. Množina všetkých návrhov tvorí pravdepodobnostný simplex v $\mathbb{R}^n$, označme ju Ξ (je kompaktná a konvexná). Označme si $M(w)$ informačnú maticu prislúchajúcu k návrhu w. Označme si $\Phi : S_+^m \to \mathbb{R} \cup \{-\infty\}$, kritérium optimality, buď D-optimalitu Φ_D alebo A-optimalitu Φ_A. Naším cieľom bude maximalizovať $\Phi(M(w))$, t.j. nájsť optimálny návrh

$$w^* = \underset{w \in \Xi}{\arg\max} \in \{\Phi(M(w))\}$$

.

2.1.1 Subspace Ascend Method

SAM algoritmus postupuje iteratívne. V každej iterácii si vyberie podpriestor v ktorom sa bude hýbať a následne spraví optimálny krok v danom podpriestore:

Algorithm 6 Subspace Ascend Method (SAM) [3]

1: Zvoľ regulárny n rozmený návrh $w^{(0)}$
2: **while** $w^{(k)}$ nespĺňa podmienky zastavenia **do**
3: Zvoľ podmnožinu bodov $S_k \subset X$
4: Nájdi aktívny podpriestor Ξ ako $\Xi_k \leftarrow \{w \in \Xi | w_x = w_x^{(k)}, x \notin S_k\}$
5: Vypočítaj $w^{(k+1)}$ ako riešenie $\max_{w \in \Xi_k} \Phi(M(w))$ spĺňajúce $\Phi(M(w^{(k+1)})) \geq \Phi(M(w^{(k)}))$
6: Set $k \leftarrow k + 1$
7: Vráť w

SAM algoritmus sa každým krokom nezmenší funkciu Φ, teda sa pohne iba smerom k optimu.

2.1.2 Vertex Exchange Method

VEM algoritmus každým krokom pomocou návrhu w zvolí nový návrh w' ako maximum návrhov $\Phi(M(w'))$ z úsečky medzi návrhom minimalizujúcom nosnú množinu a návrhom minimalizujúcim X.

Algorithm 7 Vertex Exchange Method (VEM) [3]

1: Zvoľ regulárny n rozmerný návrh w
2: **while** w nespĺňa podmienky zastavenia **do**
3: Vypočítaj $k \leftarrow \arg\min_{u \in supp(w)}\{d_u(w)\}$
4: Vypočítaj $l \leftarrow \arg\max_{v \in X}\{d_v(w)\}$
5: Vypočítaj $\alpha^* \leftarrow \arg\max_{\alpha \in [-w_l, w_k]}\{\Phi_D(M(w + \alpha e_l - \alpha e_k))\}$
6: Nastav $w_k \leftarrow w_k - \alpha^*$
7: Nastav $w_l \leftarrow w_l + \alpha^*$
8: Vráť w

Krok VEM algoritmu sa označuje ako leading Bohning exchange (ďalej LBE). Dvojica (k, l) sa označuje ako pár LBE.

2.2 Radomized Exchange Algoritmus

V tejto sekcii popíšeme randomized exchange algoritmus (REX) predstavený v [3], dá sa na neho pozerať ako na špeciálny prípad SAM algoritmu [3]. REX algoritmus kombinuje kroky VEM algoritmu a pažravých algoritmov. Podľa [3] je REX algoritmus v praxi rýchlejší ako všetky state-of-the-art algoritmy na riešenie problému optimálneho riadenia.

Nech w je regulárny návrh, nech $g(w)$ je n-rozmerný vektor s komponentami $g_x(w)$. Hlavná myšlienka REX algoritmu je počnúc inicializovaným regulárnym návrhov w iteratívne vyberať niekoľko návrhov (ich počet sa bude líšiť v rámci iterácii) a náhodne vykonať optimálnu výmenu váh medzi vybranými bodmi. Optimálna výmena váh je analogická LBE kroku VEM algoritmu. Voľba návrhov závisí na $g(w)$.

- **Krok LBE.** Pri danom návrhu w, vypočítaj $g(w)$ a urob LBE krok daný nasledovne:
$$\alpha^* \leftarrow \underset{\alpha\in[-w_l,w_k]\}}{\arg\max} \{\Phi_D(M(w+\alpha e_l-\alpha e_k)),$$
kde $k \in \arg\min_{u\in supp(w)}\{d_u(w)\}, l \in \arg\max_{v\in X}\{d_v(w)\}$. Optimálny krok $\alpha^*_{k,l}(w)$ nazvime *nulujúci*, ak je rovný buď $-w_l$ alebo w_k. To zodpovedá prípadu, keď sme sa optimálnym krokom pohli do niektorého z návrh w_l alebo w_k.

- **Výber aktívneho podpriestoru.** Podpriestor $S \subset X$, v ktorom sa pohneme bude zvolený ako zjednotenie dvoch množín. Jednou vybranou pažravým procesom (S_{greedy}) a druhou ako nosnou množinou návrh w ($S_{support}$).

  - **Pažravá množina.** Nech $L = \min(\gamma m, n)$ je počet návrhov, ktoré vyberieme. Potom zvoľ S_{greedy} ako
  $$S_{greedy} = \{l_1^*, \dots, l_L^*\} \subset X,$$
  kde l_i^* je najväčšia zložka vektoru $g(w)$.

  - **Nosná množina.** Nastav
  $$S_{support}(w) = supp(w).$$
  Označme K veľkosť nosnej množiny $K = |supp(w)|$.

  - **Aktívny podpriestor.** Aktívny podpriestor S je definovaný ako
  $$S = S_{greedy} \cup S_{support}.$$
  Váhy návrhov mimo aktívneho podpriestoru nebudú upravované v tejto iterácii.

- **Krok v aktívnom podpriestore.** Teraz vykonáme krok v ktorom aktualizujeme hodnoty w_v pre $v \in S$. Návrhy w_v pre $v \notin S$ ostanú nezmenené.

– **Tvorba párov.** Nech $(k_1, \ldots, k_K)$ je uniformne náhodná premutácia $S_{support}$ a nech $(l_1, \ldots, l_L)$ je uniformne náhodná pormutácia S_{greedy}. Potom postupnosť aktívnych návrhov je

$$(k_1, l_1), (k_2, l_1), \ldots, (k_1, l_L), (k_2, l_L), \ldots, (k_K, l_L)$$

– **Aktualizácia.** Vykonaj postupne všetky Φ-optimálne LBE kroky medzi návrhmi z $(k_1, l_1), \ldots, (k_K, l_L)$ s prisluchajúcimi aktualizáciami w a $M(w)$.

Algorithm 8 REX algoritmus [3]

Zvoľ regulárny n-rozmerný návrh w
while w nespĺňa podmienky zastavenia **do**
    Urob LBE krok vo w
    Nech k je vektor zodpovedajúci náhodnej permutácii prvkov $supp(w)$
    Nech l je vektor zodpovedajúci náhodnej permutácii $L = \min(\gamma m, n)$ indexov prvkov $g(w)$
    for $l = 1 \ldots L$ **do**
        for $l = 1 \ldots K$ **do**
            $\alpha^* \leftarrow \arg\max\{\Phi_D(M(w + \alpha e_l - \alpha e_k)) : \alpha \in [-w_l, w_k]\}$
            if LBE krok bol nulujúci alebo $\alpha^* = -w_l$ alebo $\alpha^* = w_k$ **then**
                $w_k \leftarrow w_k - \alpha^*$
                $w_l \leftarrow w_l + \alpha^*$
Vráť w

Táto sekcia je nedokončená.

Kapitola 3

Porovnanie generátorov

Táto kapitola bude obsahovať praktické porovnania algoritmov spomenutých v prvej kapitole a zdrojový kód čo najrýchlejšieho algoritmu na generovanie bodov vnútri polyédru.

Predpokladáme, že najrýchlejší generátor bude využívať MVEE elipsoid, teda táto kapitola bude obsahovať taktiež implementáciu REX algoritmu.

Táto kapitola je nedokončená.

Záver

V tejto kapitole budú zhrnuté výsledky práce. T.j. prehľad použiteľných metód spolu s ich výsledkom praktického porovnania. Okrem toho tu bude spomenutý (vlastný) prínos, čo nové daná práca prináša. Taktiež tu budú spomenuté možné smery, ktorými možno rozšíriť prácu (keďže práca má obmedzený časový rámec).

Táto kapitola je nedokončená.

Literatúra

[1] Ming-Hui Chen and Bruce W. Schmeiser. General Hit-and-Run Monte Carlo sampling for evaluating multidimensional integrals. *Operations Research Letters*, 19(4):161–169, October 1996.

[2] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, November 1995.

[3] Radoslav Harman, Lenka Filová, and Peter Richtárik. A Randomized Exchange Algorithm for Computing Optimal Approximate Designs of Experiments. *arXiv:1801.05661 [stat]*, January 2018. arXiv: 1801.05661.

[4] Radoslav Harman and Vladimír Lacko. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis*, 101(10):2297–2304, November 2010.

[5] D. J. C. Mackay. Introduction to Monte Carlo Methods. In Michael I. Jordan, editor, *Learning in Graphical Models*, NATO ASI Series, pages 175–204. Springer Netherlands, Dordrecht, 1998.

[6] Gareth O. Roberts and Jeffrey S. Rosenthal. Convergence of Slice Sampler Markov Chains. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):643–660, January 1999.