

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



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

Pod'akovanie:

Abstrakt

Klíčové slova:

Abstract

Keywords:

Obsah

Úvod	1
1 Metropolis-Hastings metódy	3
1.1 Všeobecný Metropolis-Hastings algoritmus	3
1.2 Hit-and-Run generátor	4
1.3 Gibbsov generátor	4
1.4 Slice sampling	5
2 Zamietacie metódy	7
2.1 Použitie na generovanie bodu vnútri polyédru	8
3 Randomized exchange algorithm	9
3.1 Metódy na riešenie optimal design point problému	9
3.2 Radomized Exchange Algoritmus	10

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-reprezentácia). Obidve spomenuté reprezentácie možno v prípade potreby previesť na tú druhú. Prevod medzi nimi síce nie je lacný **TODO doplniť cenu a možno spôsob**, 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 splňajú 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). **TODO doplniť, čo budeme používať**

Cieľom tejto práce je jednak poskytnúť prehľad metód na rovnomerné generovanie v polyédroch a taktiež implementovať čo najefektívnejší generátor. V prvej kapitole sa budeme zaoberať Metropolis-Hastings metódami (z triedy Markov Chain Monte Carlo), ktoré sa snažia simulovať komplexné distribúcie priamim výberom. To možno použiť aj v našom prípade, keď je cieľená distribúcia uniformná. V druhej kapitole sa budeme zaoberať zamietacími metódami, ktoré namiesto generovania bodov priamo v polyédri vygenerujú bod na jednoduchšej nadmnožine polyédra rovnomerne náhodne. Po vygenerovaní bodu overia, či leží v polyédre. Ak nie, tak generujú znovu. Tretia kapitola je venovaná algoritmu Randomized Exchange Algorithm, pomocou ktorého

možno nájsť elipsoid s minimálnym objemom obaľujúci zadaný polyéder. Tento elipsoid možno jednak priamo použiť ako nadmnožinu pri zamietacej metóde, no taktiež možno jeho osi využiť na zistenie natočenia polyédra a obalenie polyédra kvádrom s malým objemom.

TODO prechod medzi metodami

TODO motivácia za problémom - pseudonahody, vyhýbanie sa zlým príkladom,

Kapitola 1

Metropolis-Hastings metódy

V tejto kapitole sa budeme zaoberať Metropolis-Hastings algoritmom na generovanie bodov z ľubovoľnej distribúcie. Najprv sa pozrieme na všeobecný Metropolis-Hastings algoritmus, následne sa pozrieme na jeho konkrétne realizácie.

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]

```
1: inicializuj  $x^{(0)}$ 
2: for  $i = 0, 1, \dots, N$  do
3:   Vygeneruj bod  $y$  z  $Q(x^{(i)})$ 
4:   Vygeneruj  $u$  z  $U(0, 1)$ .
5:   if  $u \leq \alpha(x^{(i)}, y)$  then
6:     Nastav  $x^{(i+1)} = y$ 
7:   else
8:     Nastav  $x^{(i+1)} = x^{(i)}$ 
9: Vráť  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ .
```

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 $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ z hustoty Q .

Potrebná veľkosť N sa nazýva burn-in period. **TODO** Ake veľke N ?

TODO Všeobecnejšie Monte Carlo Markov Chain metódy - Metropolis algoritmus ako špeciálny prípad

TODO voľba parametru α

V ďalších sekciách si ukážeme niekoľko konkrétnych realizácií 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.

1.2 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 $x^{(i)}$.

Algorithm 2 Hit-and-Run generátor [1]

```

1: Inicializuj  $x^{(0)}$ 
2: for  $i = 0, \dots, N - 1$  do
3:   Vygeneruj smer  $d_i$  z distribúcie  $D$  na povrchu sféry
4:   Nájdi množinu  $S_i(d_i, x^{(i)}) = \{\lambda \in \mathbb{R}; x^{(i)} + \lambda d_i \in S\}$ 
5:   Zvoľ  $y = x^{(i)} + \lambda_i d_i$ 
6:   Vygeneruj  $u$  z  $U(0, 1)$ .
7:   if  $u \leq \alpha_i(y|x^{(i)})$  then
8:     Nastav  $x^{(i+1)} = y$ 
9:   else
10:    Nastav  $x^{(i+1)} = x^{(i)}$ 
11: Vráť  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ .
```

Nevýhodou tejto metódy je, že obsahuje pomerne veľa parametrov. Pri implementácii je potrebné vhodne zvoliť distribúciu D a parametre λ_i aj α_i .

TODO Vplyv voľby distribúcie D

TODO Dokaz konvergentnosti, rýchlosť [1]

1.3 Gibbsov generátor

V tejto sekcii 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.

Nášou úlohou je generovať z n -rozmernej distribúcie Q , pričom z Q nevieme generovať priamo. Predpokladajme, že nevieme použiť priamo Metropolis-Hastings algoritmus, lebo $Q(x^{(i)}) = Q(x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)})$ 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 $Q(x_j^{(i)} | x_1^{(i+1)}, x_2^{(i+1)} \dots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, x_{j+2}^{(i)} \dots, x_n^{(i)})$.

Gibbsov generátor bude fungovať nasledovne:

Algorithm 3 Gibbsov generátor [5]

```

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

Gibbsov generátor ako špeciálny prípad Metropolis-Hastings algoritmu má podobné vlastnosti ako Metropolis-Hastings algoritmus. Hlavnou výhodou Gibbsovho generátora oproti generátoru používajúci všeobecný Metropolis-Hastings algoritmus je, že má menej parametrov.

TODO prakticke vyuzitie

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 $\pi : \mathbb{R}^n \rightarrow [0, \infty)$, pričom $\pi(x) = \prod_{j=0}^K f_j(x)$ pre nejaké $f_j : \mathbb{R}^n \rightarrow [0, \infty)$.

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 algoritmus [6]

```

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

Daný algoritmus je závislý jedine od faktorizácie distribúcie $\pi(x)$ na $\prod_{j=0}^K f_j(x)$.

TODO rozvinut

Daný algoritmus je tiež špeciálnym prípadom Metropolis-Hastings algoritmu, možno ho analyzovať rovnakým spôsobom.

TODO porovnanie so všeobecným MH a Gibbsom

TODO využitie pri polyedroch

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

Generátor G_S vygeneruje bod $x \in S$, ak je ten bod aj z X , tak ho vráti ako výstup, inak vygeneruje nový bod $x \in S$.

Generátor G_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 vygeneruje bod z X . Platí $p_k = (1 - \frac{|X|}{|S|})^{k-1} \frac{|X|}{|S|}$. Očakávaný počet generovaní G_S je $E(G_S) = \sum_{k=0}^{\infty} k p_k = \frac{|X|}{|S|} \sum_{k=0}^{\infty} k (1 - \frac{|X|}{|S|})^{k-1} = \frac{|X|}{|S|} \frac{1}{((1 - \frac{|X|}{|S|}) - 1)^2} = \frac{|S|}{|X|}$.

Táto metóda generovania je vhodná, ak $\frac{|S|}{|X|}$ nie je veľké, t.j. ak je obal S polyédrov X dostatočne malý. Ak je $\frac{|S|}{|X|} \sim \infty$, tak je táto metóda nepoužiteľná.

2.1 Použitie na generovanie bodu vnútri polyédro

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

Žiaľ, pre takúto množinu S môže byť podiel $\frac{|S|}{|X|}$ byť ľubovoľne veľký. Ako príklad na 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 ľubovoľne veľká.

Ako ďalšia možná množina S prichádza do úvahy elipsoid obaľujúci polyéder. Keďže chceme, aby bol podiel $\frac{|S|}{|X|}$ čo najmenší, budeme skúmať elipsoid s najmenším obsahom obaľujúci polyéder - Minimum Volume Enclosing Elipsoid (ďalej MVEE). Nájsť daný elipsoid a generovať body v ňom vieme pomocou REX algoritmu, s ktorým sa oboznámime v ďalšej kapitole.

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 zobrazit elipsoid na jednotkovú guľu. Dané zobrazenie možno vypočítať pomocou osí MVEE elipsoidu.

Keďže najjednoduchšia množina S , v ktorej vieme generovať je kváder, v **TODO doplnit cislo sekcie** sa pozrieme na prípad, keď za S zvolíme kváder, ktorého osi budú zhodné s osami MVEE elipsoidu. Na daný kváder sa dá pozerat 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. Taktiež daný kváder je kváder obaľujúci MVEE elipsoid s najmenším objemom. Pre daný kváder možno spraviť odhad veľkosti: **TODO odhad veľkosti**

TODO zisti, ci je to kvader s najmensim objemom obalujuci polyeder (skor nie)

Kapitola 3

Randomized exchange algorithm

V tejto kapitole si predstavíme Randomized exchange algoritmus [3] (ďalej REX) ako metódu na riešenie optimal design problému **TODO preložiť**. Taktiež, vďaka ekvivalencii optimal design problému a minimum volume enclosing elipsoidu (MVEE) [3], sa dá využiť aj na riešenie MVEE problému. Vzhľadom na dôležitosť MVEE elipsoidu pre túto prácu a kvôli lepšiemu pochopeniu algoritmu sa najprv pozrieme na metódy riešenia optimal design problému.

3.1 Metódy na riešenie optimal design point problému

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

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 m-point design $w^{(0)}$
 - 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
-

TODO vysvetlit LBE

Algorithm 7 Vertex Exchange Method (VEM) [3]

-
- 1: Zvoľ regulárny m-point design w
 - 2: **while** $eff.act(w) < effandtime.act < time.max$ **do**
 - 3: $k \leftarrow \arg \min\{d_u(w) : u \in supp(w)\}$
 - 4: $l \leftarrow \arg \max\{d_v(w) : v \in X\}$
 - 5: $\alpha^* \leftarrow \arg \max\{\Phi_D(M(w + \alpha e_l - \alpha e_k)) : \alpha \in [-w_l, w_k]\}$
 - 6: $w_k \leftarrow w_k - \alpha^*$
 - 7: $w_l \leftarrow w_l + \alpha^*$
 - 8: Vráť w
-

3.2 Radomized Exchange Algorithmus

V tejto sekcii popíšeme randomized exchange algoritmus (REX) predstavený v [3], dá sa na neho pozeráť ako na špeciálny prípad SAM algoritmu [3].

Majme funkciu $g(x)$, ktorú chceme minimalizovať. T.j. nájsť $\min_x g(x)$. Hlavná myšlienka REX algoritmu je počnúc inicializovaným regulárnym bodom w iteratívne vyberať niekoľko bodov (ich počet sa bude líšiť v rámci iterácii) a náhodne vykonať optimálnu výmenu váh medzi vybranými bodmi. Voľba bodov závisí na $g(w)$. Optimálna výmena váh je analogická kroku VEM algoritmu.

REX algoritmus kombinuje kroky VEM algoritmu a pažravých algoritmov.

- **Krok LBE.** Pri danom bode w , vypočítaj $g(w)$ a urob LBE krok daný nasledovne:

$$\alpha^* \leftarrow \arg \max\{\Phi_D(M(w + \alpha e_l - \alpha e_k)) : \alpha \in [-w_l, w_k]\},$$

kde $k \in \arg \min\{d_u(w) : u \in supp(w)\}, l \in \arg \max\{d_v(w) : v \in X\}$. Optimálny krok $\alpha_{k,l}^*(w)$ nazvime *nulujúci*, ak je rovný buď $-w_l$ alebo w_k . To zodpovedá

prípade, keď sme sa optimálnym krokom pohli do niektorého z bodu w_l alebo w_k

TODO čomu to zodpovedá

- **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 bodu w ($S_{support}$).

- **Pažravá množina.** Nech $L = \min(\gamma m, n)$ je počet bodov, ktoré vyberieme. Potom zvol S_{greedy} ako

$$S_{greedy} = \{l_1^*, \dots, l_L^*\} \subset X,$$

kde l_i^* je najväčšia zložka vektoru $g(w)$.

- **Nosná množina.** Nastav

$$S_{support}(w) = \text{supp}(w).$$

Označme K veľkosť nosnej množiny $K = |\text{supp}(w)|$.

- **Aktívny podpriestor.** Aktívny podpriestor S je definovaný ako

$$S = S_{greedy} \cup S_{support}.$$

Váhy bodov mimo aktívneho podpriestoru nebudú upravované v tejto iterácii.

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

- **Tvorba párov.** Nech (k_1, \dots, k_K) je uniformne náhodná permutácia $S_{support}$ a nech (l_1, \dots, l_L) je uniformne náhodná permutácia S_{greedy} . Potom postupnosť aktívnych bodov je

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

- **Aktualizácia.** Vykonaj postupne všetky Φ -optimálne kroky medzi bodmi z postupnosti **TODO odkaz sa na postupnosť nad** bodov z $K \times L$ s prisluchajúcimi aktualizáciami w a $M(w)$.

Algorithm 8 REX algoritmus [3]

```

1: Zvoľ regulárny m-point design  $w$ 
2: while  $w^{(k)}$  nespĺňa podmienky zastavenia do
3:   Urob LBE krok vo  $w$ 
4:   Nech  $k$  je vektor zodpovedajúci náhodnej permutácii prvkov  $\text{supp}(w)$ 
5:   Nech  $l$  je vektor zodpovedajúci náhodnej permutácii  $L = \min(\gamma m, n)$  indexov
     prvkov  $g(w)$ 
6:   for  $l = 1 \dots L$  do
7:     for  $l = 1 \dots K$  do
8:        $\alpha^* \leftarrow \arg \max \{ \Phi_D(M(w + \alpha e_l - \alpha e_k)) : \alpha \in [-w_l, w_k] \}$ 
9:       if LBE krok bol nulujúci alebo  $\alpha^* = -w_l$  alebo  $\alpha^* = w_k$  then
10:          $w_k \leftarrow w_k - \alpha^*$ 
11:          $w_l \leftarrow w_l + \alpha^*$ 
12: Vráť  $w$ 

```

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.