Voronojevi diagrami v Hilbertovi metriki

Lucija Fekonja
lf90992@student.uni-lj.si
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija

David Lajevec
dl73333@student.uni-lj.si
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija

Dmitar Zvonimir Mitev
dm42852@student.uni-lj.si
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija

Nik Mrhar
nm88764@student.uni-lj.si
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija

POVZETEK

Hilbertova metrika je funkcija razdalje definirana za točke, ki so znotraj nekega konveksnega telesa. V članku se ukvarjamo z gradnjo Voronojevega diagrama n mest v Hilbertovi metriki. Osredotočimo se na dvorazsežni primer in na Hilbertovo metriko porojeno iz poljubnega konveksnega večkotnika z m robovi. Predstavimo dva algoritma - prvi je naključnostni, in deluje v pričakovanem času $O(mn + n \log n \log mn)$, drugi pa je deterministični, in deluje v času $O(mn \log n)$. Oba algoritma potrebujeta $O(mn)$ prostora. Prav tako pokažemo, da je kombinatorična zahtevnost Voronojevega diagrama $\Omega(mn)$. Na koncu definiramo Delaunayevo triangulacijo v Hilbertovi metriki, zapišemo nekaj njenih lastnosti in izpeljemo naključnostni algoritem za njen izračun, ki deluje v pričakovanem času $O(n(\log^3 m + \log n))$.

KEYWORDS

Hilbertova metrika, Voronojev diagram, konveksnost, simetrala, naključnostni algoritem, algoritem deli in vladaj, Delaunayeva triangulacija

1 UVOD

Hilbertovo metriko je prvi vpeljal David Hilbert leta 1895 [12] kot razdaljo med točkama v nekem konveksnem telesu $K \subset \mathbb{R}^d$. Pogosto se pojavlja pri raziskovanju konveksnih aproksimacij in ima veliko lepih lastnosti, med katerimi je tudi invarianca pri projektivnih transformacijah. Odličen vir o Hilbertovi geometriji je [15] in ga priporočamo v branje.

Znano je, da je v evklidski metriki možno izračunati Voronojev diagram n točk oz. mest v času $O(n \log n)$ s Fortuneovim algoritmom [6]. Vprašanje, na katero poskušamo odgovoriti je, kako učinkovito izračunati Voronojev diagram v Hilbertovi metriki. V ta namen predstavimo algoritma, ki rešita omenjen problem. Prvi je naključnostni prirastni in deluje v pričakovanem času $O(mn + n \log n \log mn)$. Drugi pa je deterministični deli in vladaj algoritem, ki deluje v času $O(mn \log n)$. Oba algoritma potrebujeta $O(mn)$ prostora. Izkaže se, da je kombinatorična zahtevnost Voronojevega diagrama $\Omega(mn)$, od koder sledi, da sta algoritma v najslabšem primeru do logaritemskega faktorja optimalna.

Za triangulacijo pravimo, da je Delaunayeva, če je izpolnjen Delaunayev pogoj. Ta pravi, da v nobenem očrtanem krogu trikotnikov iz triangulacije, ne sme ležati nobena točka. Koncept je predstavil Boris Delaunay leta 1934 [5]. Za izračun Delaunayeve triangulacije v evklidski metriki poznamo več algoritmov. Eden tak je naključnostni prirastni algoritem, ki deluje v povprečnem času $O(n \log n)$ in najslabšem času $O(n^2)$ [10]. Drugi znan je deterministični algoritem po principu deli in vladaj, ki prav tako deluje v času $O(n \log n)$ [7]. Ker je Delaunayeva triangulacija dualni problem Voronojevih diagramov, v kateri sta točki povezani, če sta njuni pripadajoči Voronojevi celici sosednji, jo lahko izračunamo tudi z uporabo prej predstavljenega Fortuneovega algoritma.

Sam članek je sestavljen iz osmih poglavij. V drugem poglavju vpeljemo nekatere matematične pojme povezane s tematiko. V tretjem poglavju se osredotočimo na Voronojeve diagrame v Hilbertovi metriki in zapišemo tri leme, ki jih uporabimo v poznejših dokazih. V četrtem in petem poglavju predstavimo naključnostni prirastni oz. algoritem deli in vladaj za izračun Voronojevega diagrama. Implementacijo naključnostnega algoritma si pogledamo v šestem poglavju. Nadalje, v sedmem poglavju definiramo še Delaunayeve triangulacije in izpeljemo algoritem za njen izračun. V osmem poglavju članek zaključimo.

2 OSNOVNE DEFINICIJE IN POJMI

V tem poglavju definiramo nekaj osnovnih pojmov, ki jih potrebujemo. Začnemo z definicijo *konveksnega lika.*

Definicija 1 (Konveksno telo)**.** *Konveksno telo* $K \subset \mathbb{R}^d$ je zaprta, omejena množica z neprazno notranjostjo. Njen rob označujemo z ∂K, notranjost pa z $\text{int}(K)$. Za vsako daljico, katere krajišči ležita na robu ∂K, velja, da v celoti leži v K.

V tem članku uporabljamo izraz *tetiva skozi točki p in q* za daljico, ki vsebuje obe točki in se začne in zaključi na robu telesa K. Označujemo jo s $\chi(p, q)$. Za lažje razumevanje Hilbertove metrike se najprej spomnimo *dvorazmerja* iz projektivne geometrije.

Definicija 2 (Dvorazmerje)**.** Naj bodo a, b, c, d štiri kolinearne točke v tem vrstnem redu. *Dvorazmerje* teh točk definiramo kot

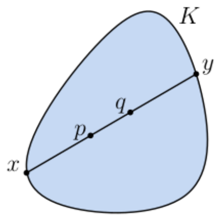
$$(a, b; c, d) = \frac{|ca|}{|cb|} \cdot \frac{|da|}{|db|}.$$

Sedaj smo pripravljeni formalno definirati Hilbertovo metriko.

Definicija 3 (Hilbertova metrika)**.** Naj bo $K \in \mathbb{R}^d$ konveksno telo. Naj bosta $p, q \in \text{int}(K), p \neq q$, in definirajmo x in y kot krajišči tetive $\chi(p, q)$, tako da so točke v vrstnem redu $\langle x, p, q, y\rangle$ (glej sliko 1). Hilbertova metrika je definirana kot:

$$H_K(p, q) = \frac{1}{2} \ln \frac{||p - y|| \cdot ||q - x||}{||q - y|| \cdot ||p - x||} = \frac{1}{2} \ln \left((p, q; y, x)\right).$$

Dodatno zahtevamo: $H_K(p, p) = 0$.



Slika 1: Hilbertova metrika.

Simetričnost in pozitivnost očitno veljata. V [17] je dokaz, da velja tudi trikotniška neenakost. Ker je Hilbertova razdalja enaka produktu logaritma dvorazmerja in pozitivne konstante, sledi, da je invariantna pri projektivnih transformacijah, saj je pri teh invariantno tudi dvorazmerje.

Definirajmo sedaj še *Hilbertove krogle*. Njihova definicija je analogna definiciji krogel v drugih metričnih prostorih. V tem članku naj bodo krogle zaprte, razen če je povedano drugače.

Definicija 4 (Hilbertova krogla)**.** Naj bo $K \subset \mathbb{R}^d$ konveksno telo. Naj bo $p \in \text{int}(K)$ in $r \geq 0$. *Hilbertova krogla* s centrom v p in radijem r je

$$B_K(p, r) = \{q \in K \mid H_K(p, q) \leq r\}.$$

V primeru, ko je $d = 2$ in je K konvenksni večkotnik z m robovi, sta Nielsen in Shao [14] pokazala, da je B_K konveksni večkotnik z največ $2m$ robovi.

3 KARAKTERISTIKE VORONOJEVIH DIAGRAMOV V HILBERTOVI METRIKI

Od zdaj naprej naj K označuje konveksen večkotnik v $\mathbb{R}^2$, in naj bo S diskretna množica točk v K, imenovanih *mesta*. Na K naj bo definirana Hilbertova metrika H_K. Vsaki točki v K lahko pripišemo mesto v S, ki ji je najbližje v smislu te metrike. Na ta način lahko za vsako točko v S definiramo *Voronojevo celico*.

Definicija 5 (Voronojeva celica)**.** *Voronojeva celica* mesta $p \in S$ glede na Hilbertovo metriko, porojeno z večkotnikom K, je

$$V_S(p) = \{q \in K \mid \forall p \in S \setminus \{p\}.\ H_K(q, p) \leq H_K(q, p')\}.$$

Voronojeve celice so *zvezde* glede na mesta p. To pomeni, da za vsako točko $q \in V_S(p)$ daljica pq v celoti leži v $V_S(p)$. Dokaz je podan v članku [8]. Skupek Voronojevih celic definira osrednjo strukturo tega članka – *Voronojev diagram*.

Definicija 6 (Voronojev diagram)**.** *Voronojev diagram* množice S v Hilbertovi metriki, porojeni z večkotnikom K, je dekompozicija ravnine glede na najbližje Voronojevo mesto. Označimo

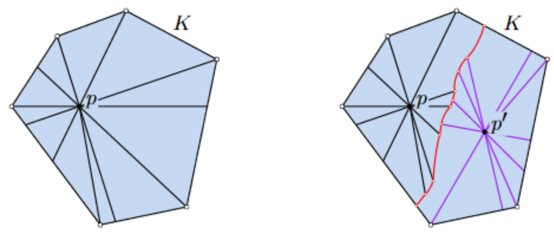
$$\text{Vor}_K(S) = (V_S(p))_{p \in S}.$$

Točka, ki ločuje eno Voronojevo celico od druge, je enako oddaljena od obeh mest, ki definirata celici. Take točke ležijo na *simetrali*.

Definicija 7 ((p, p')-simetrala)**.** Naj bosta $p, p' \in S$ mesti. Potem je (p, p')*-simetrala* množica točk

$$\{z \in K \mid H_K(z, p) = H_K(z, p')\}.$$

Naj bodo $\{v_1 \dots v_m\}$ vozlišča m-kotnika K. Potem obstaja m tetiv $\chi(v_i, p)$, ki K razdelijo na $2m$ trikotnikov. Tak trikotnik imenujemo *sektor* (glej sliko 2). Robove sektorjev, ki ležijo na (p, p')-simetrali definiramo kot *segmente* (p, p')*-simetrale*. Točke, ki ležijo med zaporednimi segmenti simetrale, imenujemo *sklepi*. Ko se premikamo vzdolž simetrale opazimo do $2m$ sektorjev celice z mestom p v cikličnem redu, in do $2m$ sektorjev celice z mestom p' v obratnem cikličnem redu. Simetrala je torej sestavljena iz največ $4m$ segmentov.



Slika 2: Sektorji celice z mestom p (levo) in (p, p')-simetrala z označenimi segmenti simetrale (desno).

Presek dveh Voronojevih celic je del simetrale, imenovan *Voronojev rob*. *Voronojevo vozlišče* je presek dveh robov. Ker je Voronojev diagram ravninski graf, z uporabo Eulerjeve formule [18] dobimo naslednjo lemo.

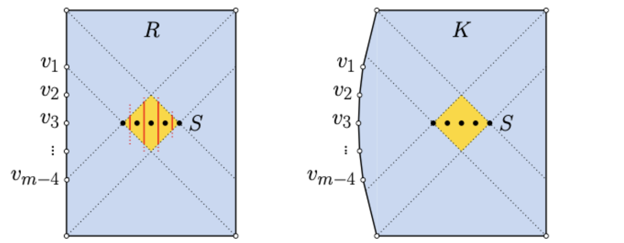
Lema 1. Za dani konveksni m-kotnik $K \subset \mathbb{R}^2$ in n mest $S \subset \text{int}(K)$, ima Voronojev diagram n celic, kvečjemu $3n$ Voronijevih robov, in kvečjemu $2n$ Voronojevih vozlišč. Vsak Voronojev rob sestoji iz največ $4m$ segmentov simetrale. Torej ima celoten diagram kombinatorično zahtevnost $O(mn)$.

Naslednja lema pove, da je kombinatorična zahtevnost $\Omega(mn)$ dosežena. Njen dokaz podaja konstrukcijo večkotnika in množice mest za katere obstaja tak Voronojev diagram.

Lema 2. Obstaja konveksni m-kotnik K in množica n mest S, $S \subset \text{int}(K)$, da ima $\text{Vor}_K(S)$ kombinatorično zahtevnost $\Omega(mn)$.

Dokaz. Naj bo R pravokotnik z vertikalno stranico nekoliko daljšo od horizontalne, S množica mest postavljenih na horizontalno črto v sredini pravokotnika in $V = \{v_1, \dots, v_{m-4}\}$ točke na levi stranici R. Položaj točk v V in S priredimo tako, da znotraj R obstaja diamanta oblika, ki vsebuje vse točke S in da se za vsak q na tem območju tetiva $\chi(q, v_i)$ seka z levo in desno stranico pravokotnika, kot prikazuje slika 3. Večkotnik K ustvarimo z ukrivljanjem leve stranice pravokotnika. Tako točke v V in ogljišča pravokotnika R postanejo ogljišča lika K.

Zaradi ukrivljanja leve stranice je med vsakim parom mest $p, p' \in S$ vsaj $m-4$ sektorjev, torej med njima obstaja simetrala, sestavljena iz vsaj $m - 4$ segmentov. Ker je parov mest natanko $n - 1$, dobimo

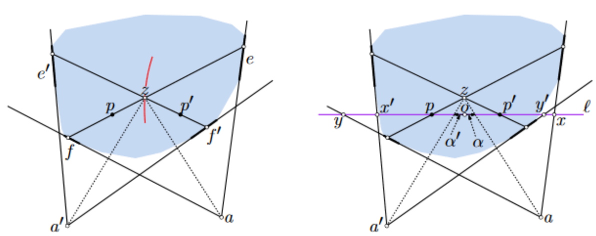


Slika 3: Pravokotnik R in večkotnik K iz dokaza leme 2.

kombinatorično zahtevnost Voronojevega diagrama $\mathrm{Vor}_K(S)$ vsaj $(m-4)(n-1) = \Omega(mn)$. *Q.E.D.*

Za konstrukcijo Voronojevega diagrama je ključno najti simetrale. Naj bo z točka na (p, p')-simetrali. Označimo z e in f točki na robu ∂K, ki se ju dotika tetiva $\chi(z, p)$. Analogno definirajmo robni točki e' in f' glede na tetivo $\chi(z, p')$, kot je prikazano na sliki 4. Izkaže se, da lahko simetralo parametriziramo s p, p', e, e', f in f'. Še več, parametrizacijo lahko prevedemo na enodimenzionalno.

Naj bo a presek tangent robov e in f in naj analog velja za a' glede na e' in f'. Če je $a = a'$ je simetrala ravna črta skozi a in z. Denimo torej, da $a \neq a'$. Naj bo ℓ premica skozi p in p'. Definirajmo točki x in y kot presek med ℓ in tangento robov e oziroma f. Podobno definiramo x' in y' za e' in f'. Ta projekcija ohrani dvorazmerje in s tem tudi Hilbertove razdalje. Naj bo o točka na ℓ, da velja enakost dvorazmerji $(o, p; y, x) = (o, p'; y', x')$, pri čemer mora o ležati med p in p'. Definirajmo še α tako, da je $o + \alpha$ točka na ℓ, v kateri premica skozi z in a seka ℓ. Analogno definirajmo α'.



Slika 4: Parametrizacija (p, p')-simetrale.

Po zgornji konstrukciji lahko vsako točko simetrale opišemo s parom (α, α'). Izkaže se, da sta tudi α in α' medsebojno odvisni po naslednji lemi, katere dokaz najdemo v dodatku [8].

Lema 3. Naj bo K konveksen večkotnik in p in p' mesti v notranjosti K. Vsaka točka na (p, p')-simetrali je lahko opisana z α in α' definiranima zgoraj. Potem obstaja linearna funkcija s parametroma $1/\alpha$ in $1/\alpha'$.

Simetralo lahko torej parametriziramo z enim od parametrov α oziroma α'.

4 NAKLJUČNOSTNI PRIRASTNI ALGORITEM

Prvi algoritem za izračun Voronojevega diagrama za množico n mest, ki so vsebovana v konveksnem m-kotniku K, je naključnostni prirastni. Algoritem sledi strukturi drugih prirastnih algoritmov za konstrukcijo Voronojevih diagramov (glej na primer [11]) in ima dodatno prednost, da generira strukturo za določanje položaja za končni diagram. Ta struktura omogoča določanje najbližjega mesta točki $q \in K$ v pričakovanem času $O((\log n)(\log mn))$, kjer je pričakovanje nad narejenimi naključnimi izbirami med konstrukcijo.

Mesta so najprej naključno permutirana in nato vstavljena ena za drugo v diagram. Osnovni primer je trivialen, saj je Voronojeva celica enega samega mesta kar celoten m-kotnik K. Da bi olajšali postopek določanja položaja, okoli vsakega mesta dodamo daljice, ki jih imenujeno *napere*. Le-te povezujejo mesta z robnimi vozlišči celice. To so: Voronojeva vozlišča, sklepi med zaporednimi simetralnimi segmenti vzdolž vsakega Voronojevega roba, presečišča Voronojevih robov in roba K ter vozlišča K, ki ležijo v tej Voronojevi celici. Ker so Voronojeve celice zvezde glede na svoje mesto, vse napere v celoti ležijo v celici. Tako dobimo topološko triangulacijo, ki jo hranimo v neki podatkovni strukturi za ravninske subdivizije kot je DCEL [2, str. 29–33].

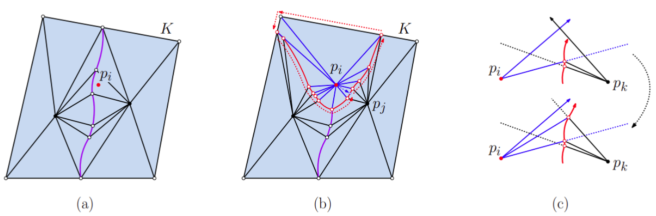
Iskanje točke v trenutnem diagramu poteka z uporabo usmerjenega acikličnega grafa, katerega začetno in končno vozlišče ustrezata trikotnikom trenutnega diagrama. Imenujemo ga *zgodovinski DAG*. Le-ta hrani zgodovino sprememb diagrama. Vstavljanje novega mesta v diagram povzroča v zgodovinskem DAG-u ustvarjanje enega ali več novih trikotnikov in odstranjevanje enega ali več starih. Vsak star trikotnik hrani tudi kazalec na nov, ki ga je zamenjal. V nadaljevanju opišemo postopek vstavljanja in določanje položaja točke.

- **Vstavljanje** poteka tako: preden vstavimo mesto p_i preiščemo zgodovinski DAG in poiščemo trikotnik, ki vsebuje p_i. Iz tega trikotnika ugotovimo najbližje mesto v trenutnem diagramu. Naj bo to p_j. Nato poiščemo sredinsko točko (v Hilbertovem smislu) daljice $p_i p_j$. Začenši v tej točki, nadaljujemo s premikanjem po robu Voronojeve celice p_i v nasprotni smeri urinega kazalca okoli p_i (glej sliko 5(b)). Možna sta dva primera, ki ju obravnavamo:
  (1) **Premikanje po simetrali** med mestoma p_i in p_k (glej sliko 5(c)). Osredotočimo se na sektorje p_i in p_k, ki vsebujejo trenunte dele simetrale. Z uporabo parametrizacije iz leme 3, lahko sledimo simetrali vse do trenutka, ko
    (i) pridemo do roba kateregakoli sektorja ali
    (ii) pridemo do simetrale med p_k in drugim mestom ali
    (iii) pridemo do roba ∂K.
    V primeru (i) ustvarimo na tem mestu novo segmentno vozlišče, dodamo napere do tega vozlišča, odstranimo nadaljevanje roba sektorja (črtkane črte na sliki 5) in nadaljujemo s sledenjem v novem sektorju. V primeru (ii) ustvarimo novo Voronejevo vozlišče, dodamo napere do tega vozlišča od treh mest, ki ga definirajo in nadaljujemo s sledenjem nove simetrale. V primeru (iii) pa dodamo naperi, ki povezujeta presečišče roba in simetrale z mesti p_i in p_k. Nato nadaljujemo sledenje po robu, kot je napisano v nadaljevanju.
  (2) **Premikanje po robu** ∂K, ki je del Voronojeve celice mesta p_i, poteka v nasprotni smeri urinega kazalca. Pred vstavljanjem p_i za vsak sektor najbližjega mesta p_k z uporabo parametrizacije iz leme 3 preverimo, ali (p_i, p_k)-simetrala seka rob ∂K znotraj presečišča trenutnih sektorjev. Če je

temu tako, na tem mestu ustvarimo vozlišče in dodamo naperi iz p_i ter p_k. Sledenje nadaljujemo po simetrali med p_i in p_k. Sicer pa je omenjeno presečišče eno od dveh robov sektorja in nadaljujemo s sledenjem v naslednjem sektorju.

Vstavljanje končamo, ko pridemo nazaj v začetno vozlišče. Na poti odstranjujemo oz. dodajemo napere z enim krajiščem v p_i.

- **Določanje položaja točke** q v zgodovinskem grafu poteka na naslednji način. Korensko vozlišče ustreza celemu večkotniku K. Vsakič, ko je trikotnik uničen zaradi novo dodane točke, shranimo kazalec na vozlišče, ki je povzročilo uničevanje. Nato v času $O(\log mn)$ poiščemo trikotnik, ki vsebuje q, tako kot to naredijo Guibas, Knuth in Sharir [11]. Naj bo p_i mesto, ki je povzročilo uničenje trikotnika, in p_j mesto, ki vsebuje q pred uničevanjem. Eno od p_i in p_j je najbližje mesto od točke q. Primerjamo razdalji med p_i in q ter p_j in q, ugotovimo katero mesto je bližje in naredimo žarično dvojiško iskanje, torej dvojiško iskanje v seznamu urejenem po kotu. Postopek zahteva $O(\log mn)$ časa.



Slika 5: Dodajanje nove točke v diagram.

Opis algoritma je na tej točki končan. V nadaljevanju analiziramo njegovo povprečno časovno zahtevnost, za katero potrebujemo dve pomožni lemi.

Lema 4. Če so mesta na začetku (enakomerno) naključno premešana, je skozi potek algoritma pričakovanih $O(mn)$ strukturnih sprememb v diagramu.

Dokaz. Dokaz poteka s povratno analizo. Imejmo Voronojev diagram n vozlišč. Ker so bila mesta na začetku premešana, je za vsako izmed njih enako verjetno, da je bilo vstavljeno zadnje. Po lemi 1 ima vsaka Voronojeva celica v povprečju $O(1)$ robov in ima vsak Voronojev rob v povprečju $O(m)$ simetral. Število strukturnih sprememb, ko je dodano zadnje mesto, je sorazmerno produktu števila Voronojevih robov v končnem diagramu in številu simetralnih segmentov vsakega od teh robov. Temu produktu dodamo še $O(m)$, ker se lahko zgodi, da Voronojeva celica vsebuje dele roba ∂K. Pričakovano število strukturnih sprememb v zadnji iteraciji vstavljanja je potem $O(1) \cdot O(m) + O(m) = O(m)$. Ker to velja ne glede na n lahko sklepamo, da je število strukturnih sprememb skozi n vstavljanj $O(mn)$. *Q.E.D.*

Lema 5. Naj bo q točka v ravnini. Če je struktura za določanje položaja točke zgrajena z vstavljanjem mest v naključnem vrstnem redu, je pričakovani čas za določanje trikotnika, ki v končnem Voronojevem diagramu vsebuje q, enak $O((\log n)(\log mn))$.

Dokaz. Trikotnik, ki vsebuje q, je odvisen od največ štirih mest. Vsak trikotnik je odvisen od mesta p_1, v katerem se njegovi dve naperi sekata. Če je osnova trikotnika simetrala, je odvisen tudi od mesta p_2 na drugi strani simetrale. Prav tako sta lahko obe preostali vozlišči p_3 in p_4, ki določata trikotnik, Voronojevi vozlišči. Torej med mesti p_1 in p_2 ter vozlišči p_3 in p_4 so največ štirje Voronojevi vozlišči. Zato je verjetnost, da se trikotnik q uniči in ustvari nov v i-ti iteraciji največ $\frac{4}{i}$. Če to seštejemo čez vse iteracije dobimo, da je pričakovana vrednost sprememb pripadajočega trikotnika q enaka $\sum_{i=1}^{n} \frac{4}{i} = 4H_n = O(\log n)$, kjer smo s H_n označili n-to harmonično število. Vsakič, ko pride do spremembe, izvedemo žarično dvojiško iskanje (obrazloženo v opisu algoritma), za kar potrebujemo $O(\log mn)$ časa. Zato je pričakovani čas iskanja enak $O(\log n \log mn)$. *Q.E.D.*

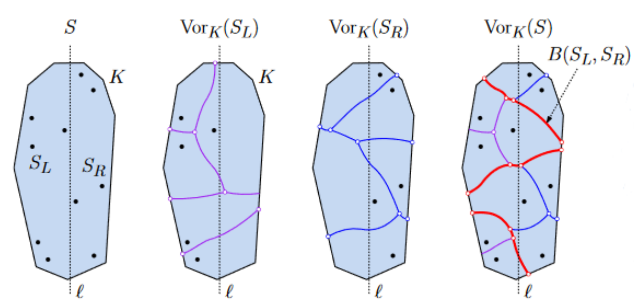
Z uporabo leme 4 in leme 5, uporabljene na vsakem izmed n mest, lahko zapišemo sledeči izrek.

Izrek 4.1. Naj bo $K \subset \mathbb{R}^2$ konveksni m-kotnik in naj bo $S \subset \mathrm{int}(K)$ množica n mest. Potem naključnostni prirastni algoritem izračuna $\mathrm{Vor}_K(S)$ v pričakovanem času $O(mn + n \log n \log mn)$, kjer je pričakovanje nad vsemi možnimi redi vstavljanja.

5 ALGORITEM DELI IN VLADAJ

V tem razdelku si ogledamo algoritem deli in vladaj za iskanje Voronojevega diagrama. Naj bo spet $K \subset \mathbb{R}^2$ konveksni m-kotnik in $S \subset \mathrm{int}(K)$ množica n mest. Struktura algoritma je standardne oblike in izvira iz algoritma v evklidski metriki, kot ga opišeta Preparata in Shamos v [16].

Problem razdelimo na manjše z delitvijo točk v S na leve S_L in desne S_R glede na navpično premico ℓ, tako da je v obeh množicah približno enako število točk. Točke delimo dokler nimata obe množici S_L in S_R zgolj ene točke. Tak primer znamo rešiti, saj Voronojev diagram z eno točko vsebuje zgolj eno Voronojevo celico, ki je enaka celotnemu večkotniku K. Nato poiščemo Voronojeva diagrama $\mathrm{Vor}(S_L)$ in $\mathrm{Vor}(S_R)$ in ju združimo z iskanjem simetral med mesti v S_L in mesti v S_R. Po lemi 3 najdemo parametrizacijo vsake simetrale. Označimo njihovo unijo z $B(S_L, S_R)$. Postopek je prikazan na sliki 6.



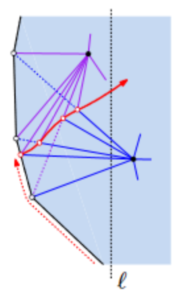
Slika 6: Iskanje Voronojevih diagramov množic S_L in S_R ter njuna združitev.

Simetrale iščemo s sprehodom od spodaj navzgor, kjer se premikamo na enega od dveh načinov.

(1) **Premikanje po simetrali**, ki razmejuje sektorja mest p_i in p_k, traja, dokler ne naletimo na enega od naslednjih primerov.

- Če simetrala seka enega od dveh robov sektorja, odstranimo nadaljevanje roba v sosednjem sektorju ter dodamo novo vozlišče, natančneje sklep, na simetrali. Slednjega z napero povežemo z mesti p_i in p_k. Premikanje nadaljujemo v novem sektorju.
- Če simetrala seka drugo simetralo, dodamo Voronojevo vozlišče in napere do vseh treh najbližjih mest. Premikanje nadaljujemo po simetrali, ki leži med mestom iz S_L in mestom iz S_R.
- Če simetrala seka rob ∂K, dodamo naperi med p_i in p_k in presečiščem ter se začnemo premikati po robu.

(2) **Premikanje po robu e lika** K vedno poteka navzgor. Torej se na robu celice z mestom iz S_R premikamo v smeri urinega kazalca, sicer pa v nasprotni smeri. Naj bosta $p_i \in S_L$ in $p_k \in S_R$ mesti najbližje opazovani točki. S parametrizacijo iz leme 3 preverimo ali (p_i, p_k)-simetrala seka rob e.
- Če najdemo presečišče, ga z naperama povežemo s p_i in p_k in se začnemo premikati po simetrali (glej sliko 7).
- Sicer naletimo na ogljišče lika K in nadaljujemo pot po sosednjem robu.



Slika 7: Iz sledenja po robu ∂K preidemo na sledenje po simetrali.

Iskanje se začne na robu ∂K v spodnjem presečišču navpične premice ℓ in lika K. Nato nadaljujemo po pravilih opisanih zgoraj dokler ne pridemo do zgornjega presečišča $\ell \cap \partial K$.

Oglejmo si še časovno zahtevnost algoritma.

Izrek 5.1. Algoritem deli in vladaj za konveksen m-kotnik K in množico n mest S izračuna $\mathrm{Vor}_K(S)$ v času $O(nm \log n)$.

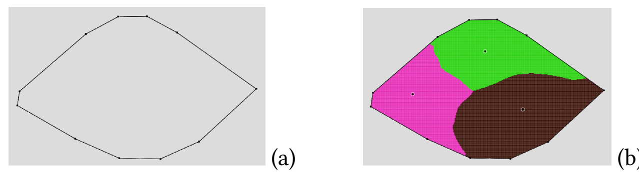
Dokaz. Algoritem problem z n mesti razdeli na dva podproblema polovične velikosti. Združevanje deluje z iskanjem simetrale, pri čemer je vsak segment simetrale najden v konstantnem času. Po lemi 1 je segmentov največ mn. Zgornji trditvi združimo in ugotovimo, da je časovna zahtevnost algoritma rešitev enačbe $T(n) = 2T(\frac{n}{2}) + mn$. Iz mojstrovega izreka sledi, da je časovna zahtevnost enaka $O(nm \log n)$. *Q.E.D.*

6 PROGRAM ZA RISANJE VORONOJEVIH DIAGRAMOV V HILBERTOVI METRIKI

V tem poglavju se osredotočimo na implementacijo naključnostnega prirastnega algoritma, ki za dana ogljišča konveksnega večkotnika in izbrana mesta vizualizira Voronojev diagram v Hilbertovi metriki. Program, ki smo ga uporabili, je bil prvotno predstavljen v članku [3] in je dostopen na portalu Github [4]. Uporabnik s klikom določi točke, ki predstavljajo konveksen večkotnik, in nato izbere mesta znotraj lika. Ob vsaki izbiri novega mesta program posodobi Voronojev diagram.

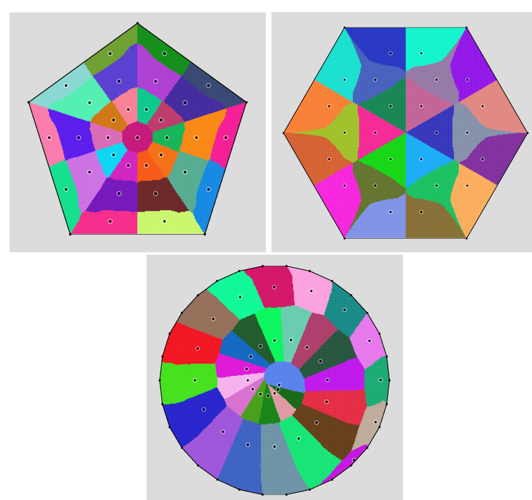
Algoritem temelji na naključnostnem prirastnem algoritmu, opisanem v poglavju 4. Recimo, da imamo konveksen večkotnik K in množico mest S. Ko uporabnik doda mesto t, algoritem razdeli K na sektorje z dodajanjem naper med t in ogljišči večkotnika. Nato za vsak $s \in S$ izračuna Voronojevo celico mesta t, označeno z $\mathrm{Vor}_{K,\{s,t\}}(t)$, pri čemer za množico mest vzame $\{s, t\}$. To stori z iskanjem simetrale, kot je opisano v 4. poglavju. Potem so Voronojeve celice mest $s \in S$ izračunane kot presek $\mathrm{Vor}_{K,S\cup\{t\}}(s) = \mathrm{Vor}_{K,S}(s) \cap \mathrm{Vor}_{K,\{s,t\}}(s)$. Na koncu doda novo celico mesta t, definirano kot $\mathrm{Vor}_{K,S\cup\{t\}}(t) = \bigcap_{s\in S} \mathrm{Vor}_{K,\{s,t\}}(t)$.

Naslednje slike prikazujejo izhod programa opisanega zgoraj.



Slika 8: (a) Izbira konveksnega večkotnika ter (b) dodajanje mest in izračun Voronojevega diagrama.

Predstavimo še nadgradnjo programa, ki prinaša boljšo uporabniško izkušnjo. Uporabniku je omogočena izbira pravilnega večkotnika, natančneje trikotnika, kvadrata, itd. Poleg tega smo dodali možnost izbire trikotniške, kvadratne in heksagonalne mreže mest, vključno z nekaj drugimi postavitvami. Prilagamo tudi nekaj slik, ki ponazarjajo delovanje programa in nove možnosti. Program je dostopen na portalu Github [13].



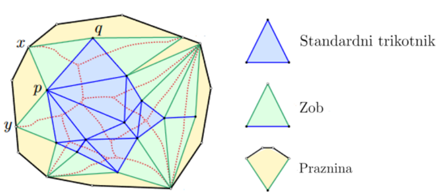
Slika 9: Pentagon s krožno mrežo mest, heksagon s heksagonalno mrežo in aproksimacija kroga s spiralno postavitvijo mest.

7 DELAUNAYEVA TRIANGULACIJA V HILBERTOVI METRIKI

Dual Voronojevega diagrama v Hilbertovi metriki, porojeni z večkotnikom K in mesti S, je *Delaunayeva triangulacija* [9], označena kot DT(S). V njej sta mesti p in q povezani, če sta njuni pripadajoči Voronojevi celici sosednji.

Povezanost dveh mest lahko preverimo s pomočjo Hilbertovih krogel. Dve mesti sta povezani natanko tedaj, ko obstaja Hilbertova krogla, katere rob vsebuje obe mesti, medtem ko njena notranjost ne vsebuje nobenega. Podobno, tri točke določajo trikotnik v Delaunayevi triangulaciji natanko tedaj, ko obstaja Hilbertova krogla, ki vsebuje vse tri točke na svojem robu in ne vsebuje nobene druge točke. Krožnico, ki omejuje to kroglo, imenujemo *Hilbertova očrtana krožnica.* Tukaj predpostavljamo splošni položaj in sicer, da nobena štiri mesta ne ležijo na robu Hilbertove krogle.

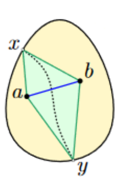
V nadaljevanju predstavimo algoritem za konstrukcijo DT(S). Za potrebe algoritma definirajmo krajišče (p, q)-simetrale kot presečišče simetrale in roba K, ki leži na levi strani usmerjene daljice pq. Prav tako definirajmo nekaj pomožnih elementov, s katerimi zagotovimo, da je K popolnoma pokrit. Trikotnik, katerega vsa oglišča so mesta, se imenuje *standardni trikotnik.* Za vsako povezavo pq, ki meji na zunanjo stran triangulacije, obstaja eno krajišče (p, q)-simetrale, ki leži na robu ∂K. Naj x označuje to krajišče. Potem trikotniku Δpqx pravimo *zob.* Vse dele večkotnika K, ki niso niti standardni trikotniki niti zobje, imenujemo *praznine.* Slednje niso nujno trikotniki, a so enolično določene s tremi točkami, zato uporabljamo notacijo Δpxy za praznino definirano z mestom p in robnima točkama x in y.



Slika 10: Standardni trikotniki, zobje in praznine.

7.1 Naključnostni prirastni algoritem

Algoritem se začne z naključno permutacijo množice S in inicializacijo triangulacije. Za slednjo izberemo poljubni mesti a in b in poiščemo krajišči (a, b)-simetrale. Nato dodamo povezavo ab in ustvarimo dva zoba tako, da povežemo a in b s krajišči simetrale, kot je prikazano na sliki 11. Postopoma dodajamo mesta in posodabljamo triangulacijo.



Slika 11: Inicializacija Delaunayeve triangulacije.

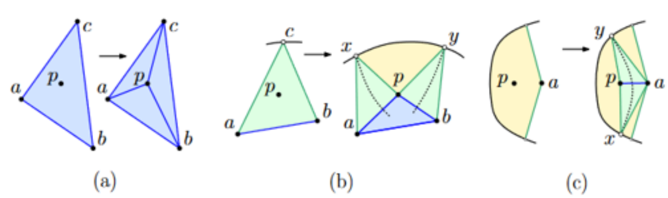
Algoritem 1 Konstrukcija Delaunayeve triangulacije množice S.

Procedura DELAUNAYEVA TRIANGULACIJA(S)
  $\tau \leftarrow$ prazna triangulacija
  Naključno permutiraj S
  $a, b \leftarrow$ poljubni točki iz S
  $x, y \leftarrow$ krajišči (a, b)-simetrale
  Dodaj povezave ab, ax, ay, bx in by v τ
  for all $p \in S \setminus \{a, b\}$ **do**
    DODAJ(p, τ)
  end for
  return τ
end Procedura

Vsako naslednjo točko lahko dodamo v standardni trikotnik, zob ali praznino. Če jo dodamo v standardni trikotnik, jo povežemo z njegovimi ogljišči. Če točko dodamo v zob, izbrišemo ogljišče, ki je na robu, povežemo novo točko z najbližjima mestoma in dodamo dva nova zoba glede na simetrali med mestoma in novo točko. Če pa točko dodamo v praznino, jo povežemo z edinim mestom v praznini in dodamo dva nova zoba. Vsi trije primeri so prikazani na sliki 12.

Algoritem 2 Dodajanje točke p iz S v triangulacijo τ.

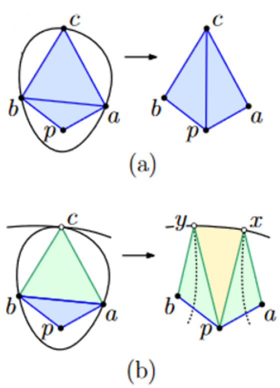
Procedura DODAJ(p, τ)
  $\Delta abc \leftarrow$ trikotnik v τ, ki vsebuje p
  if Δabc je standardni trikotnik **then**
    Dodaj povezave ap, bp in cp v τ
    OBRNI POVEZAVO(ab, p, τ), OBRNI POVEZAVO(bc, p, τ),
    OBRNI POVEZAVO(ca, p, τ)
  else if Δabc je zob **then**
    Naj bosta a in b mesti in c krajišče (a, b)-simetrale
    $x, y \leftarrow$ krajišči (a, p)- oz. (p, b)-simetral
    Iz τ odstrani vozlišče c ter povezavi ac in bc
    Dodaj povezave ap, bp, ax, px, by in py v τ
    OBRNI POVEZAVO(ab, p, τ)
    POPRAVI ZOB($\Delta apx, p, \tau$), POPRAVI ZOB($\Delta pby, p, \tau$)
  else
    Naj bo a mesto in naj bosta b in c na robu
    $x, y \leftarrow$ krajišči (a, p)- in (p, a)-simetral
    Dodaj povezave ap, ax, px, ay in py v τ
    POPRAVI ZOB($\Delta apx, p, \tau$), POPRAVI ZOB($\Delta pay, p, \tau$)
  end if
end Procedura



Slika 12: Dodajanje v (a) standardni trikotnik, (b) zob in (c) praznino.

Ob dodajanju novega vozlišča se lahko zgodi, da nova triangulacija ne ustreza Delaunayevemu pogoju. V nekaterih primerih pa lahko celo pride do izgube geometrijske pravilnosti, na primer, ko se dva zoba prekrivata. Morebitne kršitve popravimo s procedurama Obrni povezavo in Popravi zob.

Procedura Obrni povezavo sprejme usmerjeno povezavo ab in novo dodano točko p, ki leži na levi strani ab. Nato poišče trikotnik Δabc, ki leži na desni strani povezave ab, in preveri ali p leži znotraj očrtane krožnice določene s točkami a, b in c. Če to velja, odstrani povezavo ab in preveri ali je Δabc standardni trikotnik ali zob. V kolikor je standardni trikotnik, doda povezavo pc in preveri ali morata biti tudi povezavi ac in cb obrnjeni. Če je Δabc zob, odstrani vozlišče c na robu, poišče krajišči x in y (p, a)- in (b, p)-simetral in v τ doda nova zoba Δpax in Δbpy.

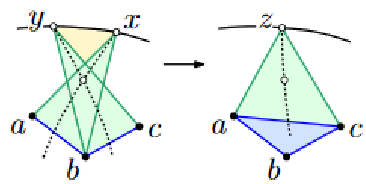


Slika 13: Obračanje povezave v (a) standardnem trikotniku in (b) zobu.

Algoritem 3 Obračanje povezav, v primeru kršenja pogoja praznih hilbertovih krogov.

Procedura Obrni povezavo(ab, p, τ)
  $c \leftarrow$ vozlišče trikotnika na desni strani povezave ab v τ
  if p leži v Hilbertovem očrtanem krogu določenem z a, b, c **then**
    Odstrani povezavo ab iz τ
    if Δabc je standardni trikotnik **then**
      Dodaj povezavo pc v τ
      Obrni povezavo(ac, p, τ), Obrni povezavo(cb, p, τ)
    else
      $x, y \leftarrow$ krajišči (p, a)- in (b, p)-simetral
      Odstrani vozlišče c in povezavi ac ter bc it τ
      Dodaj povezave ax, px, by in py v τ
    end if
  end if
end Procedura

Procedura Popravi zob razreši probleme, ko se dva zoba prekrivata. Procedura sprejme trikotnik Δabx, kjer sta a in b mesti, x pa krajišče na robu. Novo dodano mesto je ena izmed točk a ali b. Naj bo Δbcy sosednji zob zoba Δabx v smeri urinega kazalca. Če se prekrivata, ju zamenjamo s standardnim trikotnikom Δabc in zobom Δacz, kjer je z krajišče (a, c)-simetrale. Če je novo dodana točka a, moramo preveriti ali je potrebno obrniti povezavo bc in popraviti zob Δacz. Isto proceduro ponovimo še na zobu, ki je sosednji Δabx v nasprotni smeri urinega kazalca.



Slika 14: Popravljanje prekrivajočih se zob.

Algoritem 4 Popravljanje zob, ki se prekrivajo.

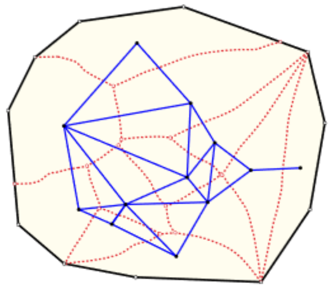
Procedura Popravi zob$(\Delta abx, p, \tau)$
  $\Delta bcy \leftarrow$ zob sosednji zobu Δabx v smeri urinega kazalca
  if Δabx in Δbcy se prekrivata **then**
    $z \leftarrow$ krajišče (a, c)-simetrale
    Odstrani vozlišči x in y ter povezave ax, bx, by in cy iz τ
    Dodaj povezave ac, az in cz v τ
    if $p = a$ **then**
      Obrni povezavo(bc, p, τ)
      Popravi zob$(\Delta acz, p, \tau)$
    end if
  else
    Ponovi za zob, ki je sosednji Δabx v nasprotni smeri urinega kazalca
  end if
end Procedura

Na koncu izpeljimo pričakovano časovno zahtevnost danega algoritma. Podobno kot pri Delaunayevi triangulaciji v Evklidskem prostoru, lahko določimo trikotnik, v katerem je vsebovana novo dodana točka v pričakovanem času $O(\log n)$ z definiranjem podatkovne strukture za določanje položaja točke, podobno kot to storijo v [11]. Glavna razlika v časovni zahtevnosti nastane zaradi izračuna Hilbertovih očrtanih krožnic. Izkaže se, da se le-te lahko izračunajo v pričakovanem času $O(\log^3 m)$ [9]. Faktorja $O(\log n)$ in $O(\log^3 m)$ se izvedeta konstantno mnogokrat za vsako od n dodanih točk. Tako smo dokazali naslednjo trditev.

Trditev 1. Naključnostna prirastna konstrukcija Delaunayeve triangulacije množice n točk znotraj konveksnega m-kotnika ima pričakovano časovno zahtevnost $O(n(\log^3 m + \log n))$.

7.2 Dualnost Delaunayeve triangulacije in Voronojevega diagrama

Kot je bilo nakazano v uvodu tega poglavja, sta Delaunayeva triangulacija in Voronojev diagram povezana kot dualna pojma. To pomeni, da lahko iz enega izpeljemo drugega. V tem poglavju predstavimo to povezavo in pri tem sledimo konstrukciji, kot je bila izvedena v evklidski metriki [1].



Slika 15: Dualnost Voronojevega diagrama in Delaunajve triangulacije.

Delaunayevo triangulacijo lahko konstruiramo iz Voronojevega diagrama tako, da za vsak Voronojev rob določimo dve točki (mesti), katerih Voronojevi celici sta ločeni s tem robom. Povezava med tema dvema točkama postane rob trikotnika v Delaunayevi triangulaciji. Ko pregledamo vse Voronojeve robove, je konstrukcija končana.

Robovi Voronojevega diagrama se določijo iz samega diagrama, ki ga konstruiramo z uporabo enega od prej opisanih algoritmov. Če se odločimo za uporabo naključnostnega prirastnega algoritma, lahko diagram izračunamo v pričakovanem času $O(mn{+}n \log n \log mn)$. Nato za vsak Voronoijev rob poiščemo dve mesti, katerih Voronojevi celici sta ločeni s tem robom. To lahko storimo v konstantnem času. Ker je Voronojevih robov kvečemu $3n$, kot trdi lema 1, je časovna zahtevnost iskanja parov mest enaka $O(n)$. Skupna časovna zahtevnost izračuna Delaunayeve triangulacije iz Voronojevega diagrama je torej $O(mn + n \log n \log mn) + O(n) = O(mn + n \log n \log mn)$.

Voronojev diagram je mogoče zgraditi iz Delaunayeve triangulacije na naslednji način. Najprej poiščemo trikotnike v Delaunayeve triangulaciji, nato za vsak trikotnik določimo središče njemu očrtane krožnice v smislu Hilbertove metrike. Nazadnje konstruiramo simetrale med točkami v Delaunayevi triangulaciji. Za povezavo $p_i p_j$ v Delaunayevi triangulaciji, ki si jo delita trikotnika $\Delta p_i p_j p_k$ in $\Delta p_i p_l p_j$, s parametrizacijo iz leme 3 konstruiramo (p_i, p_j)-simetralo. Le-ta se mora začeti in končati v središčih očrtanih krožnic omenjenih trikotnikov. Ko preverimo vse povezave v Delaunayevi triangulaciji, končamo s konstrukcijo Voronojevega diagrama.

Trikotnike Delaunayeve triangulacije določimo iz triangulacije, ki jo z naključnostnim prirastnim algoritmom izračunamo v času $O(n(\log^3 m + \log n))$, kot navaja trditev 1.

V drugem koraku konstrukcije moramo izračunati središča očrtanih krožnic vseh trikotnikov v Delaunayevi triangulaciji. Za posamezen trikotnik se središče izračuna na naslednji način. Za vsak par robnih točk izračunamo presečišče premice, ki poteka skozi te točke, z robom mnogokotnika. Če ima slednji m robov, lahko to storimo v času $O(m)$. Ti dve presečišči uporabimo za izračun Hilbertove razdalje med oglišči mnogokotnika, kar traja $O(1)$. Nato izberemo začetno aproksimacijo središča očrtanega trikotnika in izračunamo razdalje do vsakega od oglišč trikotnika, da posodobimo položaj središča. Če je potrebnih k korakov za dosego določene tolerance enakosti razdalj od središča do oglišč, je časovna zahtevnost iskanja središč enaka $O(km)$. Ker je vseh trikotnikov $O(n)$, iskanje središč očrtanih trikotnikov zahteva $O(kmn)$ časa.

Na koncu algoritem poišče simetralo med vsakim parom n točk. V članku [9] so Gezalyan in drugi izpeljali, da je posamezno simetralo možno konstruirati v času $O(\log^2 m)$. Torej je za vse pare mest, ki so povezani v Delaunajevi triangulaciji, možno izračunati simetrale v času $O(n \log^2 m)$. Zaključimo, da konstrukcija Voronojevega diagrama iz Delaunajeve triangulacije zahteva $O(n(\log^3 m + \log n) + kmn + n \log^2 m)$ časa.

8 ZAKLJUČEK

Predstavljena sta bila dva učinkovita algoritma za izračun Voronojevega diagrama v Hilbertovi metriki: naključnostni prirastni algoritem in deterministični algoritem deli in vladaj. Prvi ima pričakovano časovno zahtevnost $O(mn + n \log n \log mn)$, drugi pa deterministično časovno zahtevnost $O(mn \log n)$. Oba algorima potrebujeta prostorsko zahtevnost $O(mn)$. Dokazana je tudi kombinatorična optimalnost Voronojevega diagrama v Hilbertovi metriki, $\Theta(mn)$. To algoritma uvršča med optimalne v najslabšem primeru do logaritemskega faktorja. Poleg tega je bil predstavljen algoritem za izračun Delaunayeve triangulacije v obravnavani metriki.

Prihodnje raziskave se lahko osredotočijo na razširitev algoritmov na višje razsežnosti, proučevanje vpliva različnih konveksnih likov ter nadaljnje izboljšanje učinkovitosti in primerjalno analizo obeh algoritmov.

LITERATURA

[1] Algorithms for Competitive Programming delaunay triangulation and voronoi diagram. https://cp-algorithms.com/geometry/delaunay.html. Dostopano: 2024-05-18.

[2] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[3] Madeline Bumpus, Caesar Dai, Auguste H Gezalyan, Sam Munoz, Renita Santhoshkumar, Songyu Ye, and David M Mount. Software and analysis for dynamic voronoi diagrams in the hilbert metric. *arXiv preprint arXiv:2304.02745*, 2023.

[4] Xufeng Dai. Dynamic voronoi diagrams in the hilbert metric. https://github.com/caesardai/Voronoi_In_Hilbert/tree/main.

[5] Boris Delaunay. Sur la sphère vide. a la mémoire de georges voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 6: 793–800, 1934.

[6] Steven Fortune. A sweepline algorithm for voronoi diagrams. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, page 313–322. Association for Computing Machinery, 1986.

[7] Steven Fortune and Christopher J. Van Wyk. Efficient exact arithmetic for computational geometry. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, SCG '93, page 163–172, New York, NY, USA, 1993. Association for Computing Machinery.

[8] Auguste Gezalyan and David Mount. Voronoi diagrams in the hilbert metric. In *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258, pages 35:1–35:16, Dagstuhl, Germany, 2023.

[9] Auguste Gezalyan, Soo Kim, Carlos Lopez, Daniel Skora, Zofia Stefankovic, and David M Mount. Delaunay triangulations in the hilbert metric. *arXiv preprint arXiv:2312.05987*, 2023.

[10] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2):74–123, apr 1985.

[11] Leonidas Guibas, Donald Knuth, and Micha Sharir. Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(1–6):381–413, 6 1992.

[12] David Hilbert. Ueber die gerade linie als kürzeste verbindung zweier punkte. *Mathematische Annalen*, 46:91–96, 1895.

[13] Nik Mrhar. Implementacija algoritma za voronojev diagram v hilbertovi metriki. https://github.com/MrharN18/Voronoi/tree/main.

[14] Frank Nielsen and Laetitia Shao. On Balls in a Hilbert Polygonal Geometry. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77, pages 67:1–67:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

[15] Athanase Papadopoulos and Marc Troyanov. *Handbook of Hilbert geometry*, volume 22 of *IRMA Lectures in Mathematics and Theoretical Physics.* European Mathematical Society, 2014.

[16] Franco Preparata and Michael Shamos. *Computational Geometry: an Introduction.* Springer, 1985.

[17] Sumio Yamada. Convex bodies in euclidean and weil-petersson geometries. *Proceedings of the American Mathematical Society*, 142, 10 2011.

[18] Mitsuharu Yamamoto, Shin-ya Nishizaki, Masami Hagiya, and Yozo Toda. Formalization of planar graphs. In *Higher Order Logic Theorem Proving and Its Applications: 8th International Workshop Aspen Grove, UT, USA, September 11–14, 1995 Proceedings 8*, pages 369–384. Springer, 1995.