Voronojevi diagrami v Hilbertovi metriki

Lucija Fekonja, David Lajevec, Dmitar Zvonimir Mitev, Nik Mrhar

Fakulteta za matematiko in fiziko
Univerza v Ljubljani

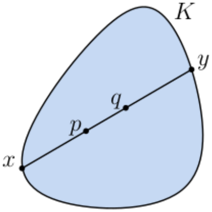
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Agenda

- Hilbertova metrika
- Naključnostni prirastni algoritem
- Algoritem deli in vladaj
- Programska oprema za risanje VD
- Delaunayeva triangulacija v Hilbertovi metriki

Hilbertova metrika

- $H_K(p, q) = \frac{1}{2} \ln \frac{||p-y|| \cdot ||q-x||}{||q-y|| \cdot ||p-x||} = \frac{1}{2} \ln(p, q; y, x);$
- $H_K(p, p) = 0.$

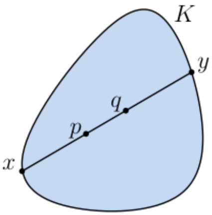


Slika: Hilbertova metrika

- invariantna na projektivne transformacije

Hilbertova metrika

- $H_K(p,q) = \frac{1}{2}\ln\frac{||p-y||\cdot||q-x||}{||q-y||\cdot||p-x||} = \frac{1}{2}\ln(p,q;y,x)$;
- $H_K(p,p) = 0$.



Slika: Hilbertova metrika

- invariantna na projektivne transformacije
- omejimo se na primer konveksnega večkotnika

Naključnostni prirastni algoritem

- Naj bo $K \subset \mathbb{R}^2$ koveksni m-kotnik in $S \subset \mathbb{R}^2$ množica $(n+1)$-točk. Naš cilj je izračunati $\mathrm{Vor}_K(S)$ in zgraditi strukturo za določanje položaja poljubne točke $q \in \mathbb{R}^2$.

Ideja algoritma

1. naključno permutiraj množico točk $S = \{p_0, p_1, ..., p_n\}$;

Ideja algoritma

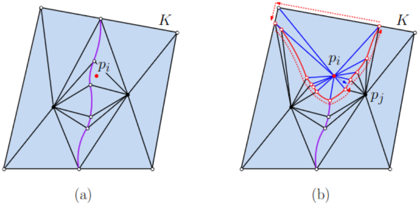
1. naključno permutiraj množico točk $S = \{p_0, p_1, ..., p_n\}$;
2. zgradi $\mathrm{VD}(\{p_0\}) \equiv K$;

Ideja algoritma

1. naključno permutiraj množico točk $S = \{p_0, p_1, ..., p_n\}$;
2. zgradi $\mathrm{VD}(\{p_0\}) \equiv K$;
3. za $i \in \{1, \ldots, n\}$:
   - iz $\mathrm{VD}(\{p_0, \ldots, p_{i-1}\})$ zgradi $\mathrm{VD}(\{p_0, \ldots, p_{i-1}, p_i\})$.

Iterativno vstavljanje točk

Uporabljamo t. i. zgodovinski DAG, ki hrani trikotnike trenutnega diagrama.

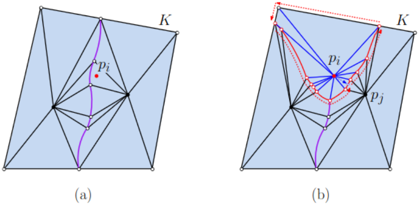


Slika: vstavljanje novega mesta

1. poiščemo trikotnik, ki vsebuje točko p_i, ki jo želimo vstaviti;

Iterativno vstavljanje točk

Uporabljamo t. i. zgodovinski DAG, ki hrani trikotnike trenutnega diagrama.

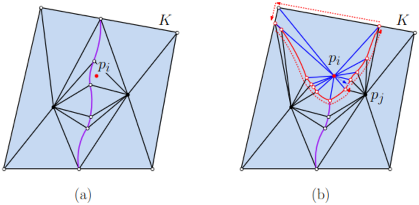


Slika: vstavljanje novega mesta

1. poiščemo trikotnik, ki vsebuje točko p_i, ki jo želimo vstaviti;
2. poiščemo točko p_j, ki je najbližja točki p_i;

Iterativno vstavljanje točk

Uporabljamo t. i. zgodovinski DAG, ki hrani trikotnike trenutnega diagrama.

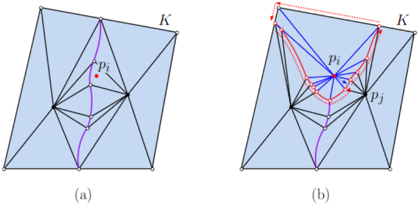


Slika: vstavljanje novega mesta

1. poiščemo trikotnik, ki vsebuje točko p_i, ki jo želimo vstaviti;
2. poiščemo točko p_j, ki je najbližja točki p_i;
3. začnemo v sredinski točki daljice $p_i p_j$ in sledimo robu Voronojeve celice v nasprotni smeri urinega kazalca;

Iterativno vstavljanje točk

Uporabljamo t. i. zgodovinski DAG, ki hrani trikotnike trenutnega diagrama.

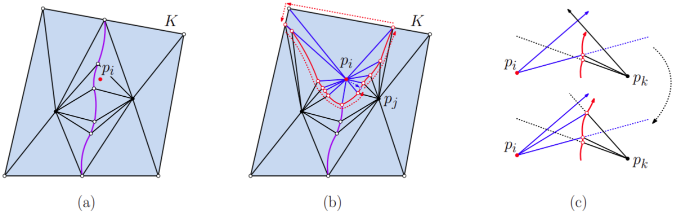


Slika: vstavljanje novega mesta

1. poiščemo trikotnik, ki vsebuje točko p_i, ki jo želimo vstaviti;
2. poiščemo točko p_j, ki je najbližja točki p_i;
3. začnemo v sredinski točki daljice $p_i p_j$ in sledimo robu Voronojeve celice v nasprotni smeri urinega kazalca;
4. na poti uničujemo trikotnike in gradimo nove.

Sledenje robov – po simetrali točk ali po robu K

- obravnavamo tri (štiri) primere v odvisnosti od točke, ki jo srečamo:

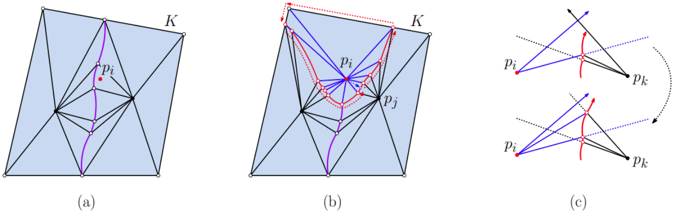


Slika: vstavljanje nove točke

1. če pridemo do roba sektorja ...

Sledenje robov – po simetrali točk ali po robu K

- obravnavamo tri (štiri) primere v odvisnosti od točke, ki jo srečamo:

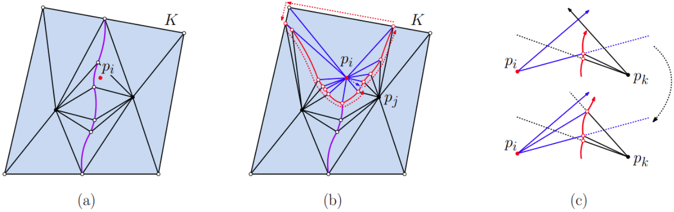


Slika: vstavljanje nove točke

1. če pridemo do roba sektorja ...
2. če pridemo do simetrale med p_i in drugim mestom ...

Sledenje robov – po simetrali točk ali po robu K

- obravnavamo tri (štiri) primere v odvisnosti od točke, ki jo srečamo:



Slika: vstavljanje nove točke

1. če pridemo do roba sektorja ...
2. če pridemo do simetrale med p_i in drugim mestom ...
3. če pridemo do roba K, nadaljujemo sledenje po robu ali po simetrali ...

Naključ. prirastni algoritem – časovna zahtevnost

Izkaže se, da je:

Naključ. prirastni algoritem – časovna zahtevnost

Izkaže se, da je:

1. $O(mn)$ pričakovano število strukturnih sprememb v diagramu čez vse iteracije (povratna analiza);

Naključ. prirastni algoritem – časovna zahtevnost

Izkaže se, da je:

1. $O(mn)$ pričakovano število strukturnih sprememb v diagramu čez vse iteracije (povratna analiza);
2. $O(\log n \log mn)$ pričakovano število operacij za določanje trikotnika, ki v zgodovinskem DAG-u vsebuje dano točko.

Naključ. prirastni algoritem – časovna zahtevnost

Izkaže se, da je:

1. $O(mn)$ pričakovano število strukturnih sprememb v diagramu čez vse iteracije (povratna analiza);
2. $O(\log n \log mn)$ pričakovano število operacij za določanje trikotnika, ki v zgodovinskem DAG-u vsebuje dano točko.

- algoritem izračuna $\mathrm{Vor}_K(S)$ v pričakovanem času $O(mn + n \log n \log mn)$;

Naključ. prirastni algoritem – časovna zahtevnost

Izkaže se, da je:

1. $O(mn)$ pričakovano število strukturnih sprememb v diagramu čez vse iteracije (povratna analiza);
2. $O(\log n \log mn)$ pričakovano število operacij za določanje trikotnika, ki v zgodovinskem DAG-u vsebuje dano točko.

- algoritem izračuna $\mathrm{Vor}_K(S)$ v pričakovanem času $O(mn + n \log n \log mn)$;
- Je do logaritemskega faktorja optimalen, saj je kombinatorična kompleksnost $\mathrm{Vor}_K(S) = \Omega(mn)$;

Algoritem deli in vladaj

- izračun Voronojevega diagrama za n točk v konveksnem m-kotniku $K \subset \mathbb{R}^2$ v pričakovanem času $O(nm \log n)$

Glavni koraki algoritma

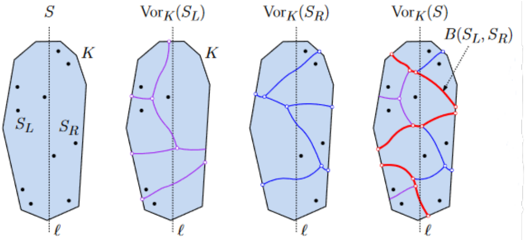
1. Razdelitev točk S na leve S_L in desne S_R glede na navpično premico ℓ

Glavni koraki algoritma

1. Razdelitev točk S na leve S_L in desne S_R glede na navpično premico ℓ
2. Rekurzivno delimo točke, dokler obe množici S_L in S_R ne vsebujeta zgolj ene točke

Glavni koraki algoritma

1. Razdelitev točk S na leve S_L in desne S_R glede na navpično premico ℓ
2. Rekurzivno delimo točke, dokler obe množici S_L in S_R ne vsebujeta zgolj ene točke
3. Združimo Voronojeva diagrama $\text{Vor}(S_L)$ in $\text{Vor}(S_R)$ z iskanjem simetral med točkami v S_L in S_R



Slika: Algoritem deli in vladaj

Iskanje simetral

1. Premikanje po simetrali

Iskanje simetral

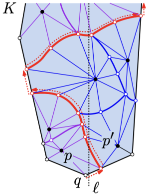
1. Premikanje po simetrali
   - Simetrala seka rob sektorja

Iskanje simetral

1. Premikanje po simetrali
   - Simetrala seka rob sektorja
   - Simetrala seka drugo simetralo

Iskanje simetral

1. Premikanje po simetrali
   - Simetrala seka rob sektorja
   - Simetrala seka drugo simetralo
   - Simetrala seka ∂K



Iskanje simetral

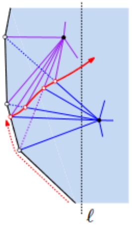
2. Premikanje po ∂K
   - Premikamo se v smeri urinega kazalca (mesto iz S_R) ali v nasprotni smeri (mesto iz S_L)

Iskanje simetral

2. Premikanje po ∂K
   - Premikamo se v smeri urinega kazalca (mesto iz S_R) ali v nasprotni smeri (mesto iz S_L)
   - Preverimo, ali simetrala med p_i in p_k seka rob

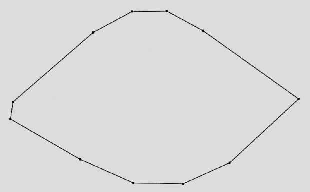
Iskanje simetral

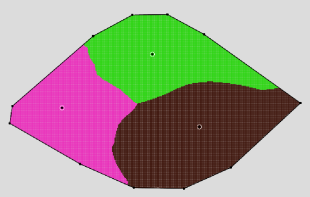
2. Premikanje po ∂K
   - Premikamo se v smeri urinega kazalca (mesto iz S_R) ali v nasprotni smeri (mesto iz S_L)
   - Preverimo, ali simetrala med p_i in p_k seka rob
   - Če ni presečišča, nadaljujemo pot po sosednjem robu



Programska oprema

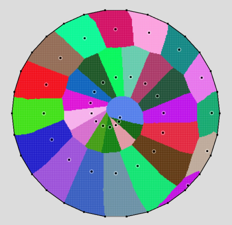
- implementacija temelji na naključnostnem prirastnem algoritmu
- uporabnik določi konveksen večkotnik K
- z vsakim novo dodanim mestom, se Voronojev diagram posodablja

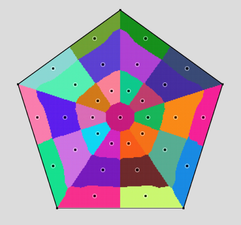


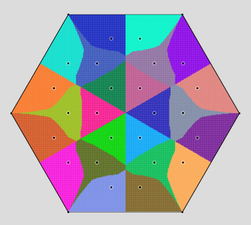


Nadgradnja programske opreme

- možnost izbire različnih večkotnikov
- možnost izbire različnih mrež mest

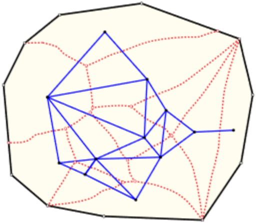






Delaunayjeva triangulacija kot dual

- vozlišča v DT predstavljajo mesta Voronojevih celic
- povezave v DT povezujejo sosednje Voronojeve celice
- središča očrtanih krožnic v DT predstavljajo Voronojeva vozlišča



Slika: Dualnost Delaunayeve triangulacije in Voronojevega diagrama.

Izračun Delaunayje triangulacije v Hilbertovi metriki

Trditev

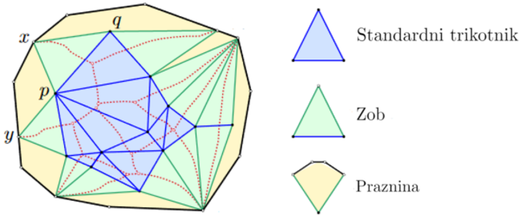
Naključnostna prirastna konstrukcija Delaunayeve triangulacije množice n točk znotraj konveksnega m-kotnika ima pričakovano časovno zahtevnost $O(n(\log^3 m + \log n))$.

Naključnostni prirastni algoritem za DT

- **sprejme** točke znotraj konveksnega večkotnika, na katerem je definirana Hilbertova metrika
- **vrne** Delaunayevo triangulacijo danih točk

Naključnostni prirastni algoritem za DT

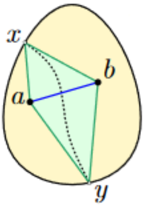
- **sprejme** točke znotraj konveksnega večkotnika, na katerem je definirana Hilbertova metrika
- **vrne** Delaunayevo triangulacijo danih točk



Slika: Pomožni konstrukti.

Naključnostni prirastni algoritem za DT

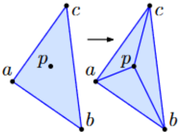
1. inicializacija triangulacije



Naključnostni prirastni algoritem za DT

3. postopoma dodajamo točke in posodabljamo Delaunayevo triangulacijo

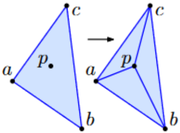
- v standardni trikotnik:



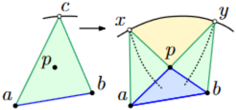
Naključnostni prirastni algoritem za DT

3. postopoma dodajamo točke in posodabljamo Delaunayevo triangulacijo

- v standardni trikotnik:



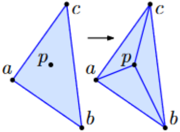
- v zob:



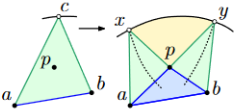
Naključnostni prirastni algoritem za DT

3. postopoma dodajamo točke in posodabljamo Delaunayevo triangulacijo

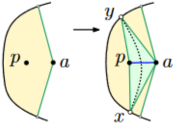
- v standardni trikotnik:



- v zob:

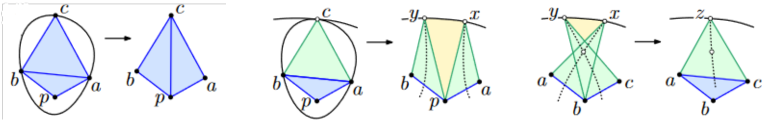


- v praznino:



Naključnostni prirastni algoritem za DT

2. preverimo pravilnost triangulacije
    - veljati mora Delaunayev pogoj
    - zobje se ne smejo prekrivari



THANK YOU