Algoritmi u geometriji: priča sa srećnim krajem

Nikola Jovanović

Matematička gimnazija

NEDELJA INFORMATIKE v5.0

19. decembar 2018.

Uvod

Cilj

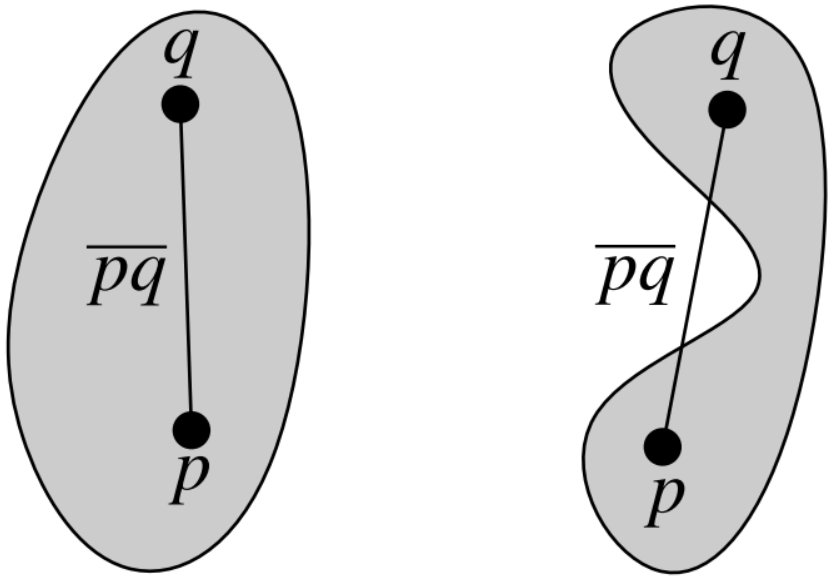
- **Bavićemo se:**
  - Algoritmima u geometriji
  - Realnim problemima koji ih inspirišu
- **Nećemo se baviti:**
  - Implementacionim detaljima
  - Graničnim slučajevima
- **Napomena**:
  - Animacije ~~rade samo u *Acrobat Reader*-u~~ ne rade nigde :(
  - Naslov svake animacije je link ka izvoru, ako još uvek postoji

Pregled

- Konveksni omotač
- Voronoi dijagram
- *Honourable mentions*

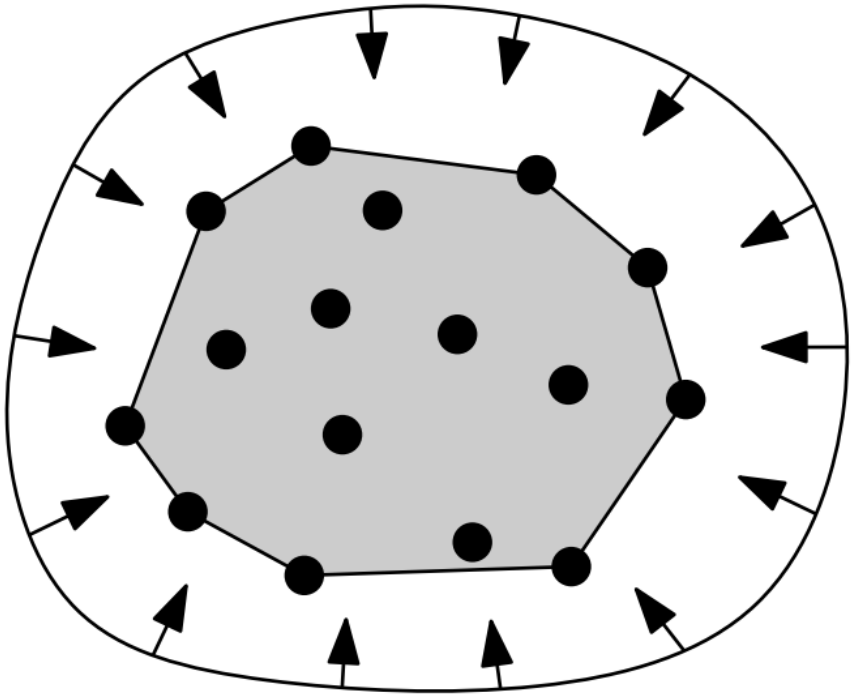
Konveksni omotač

Konveksnost



- Skup tačaka S je **konveksan** ako je za svaki par tačaka p i q iz S cela duž $\overline{pq}$ unutar S
- **Konveksna kombinacija** tačaka $x_1, x_2, \ldots, x_n$ je svaka tačka oblika $a_1x_1 + a_2x_2 + \cdots + a_nx_n$ tako da važi $\forall a_i : a_i \geq 0$ i $a_1 + a_2 + \cdots + a_n = 1$

Konveksni omotač

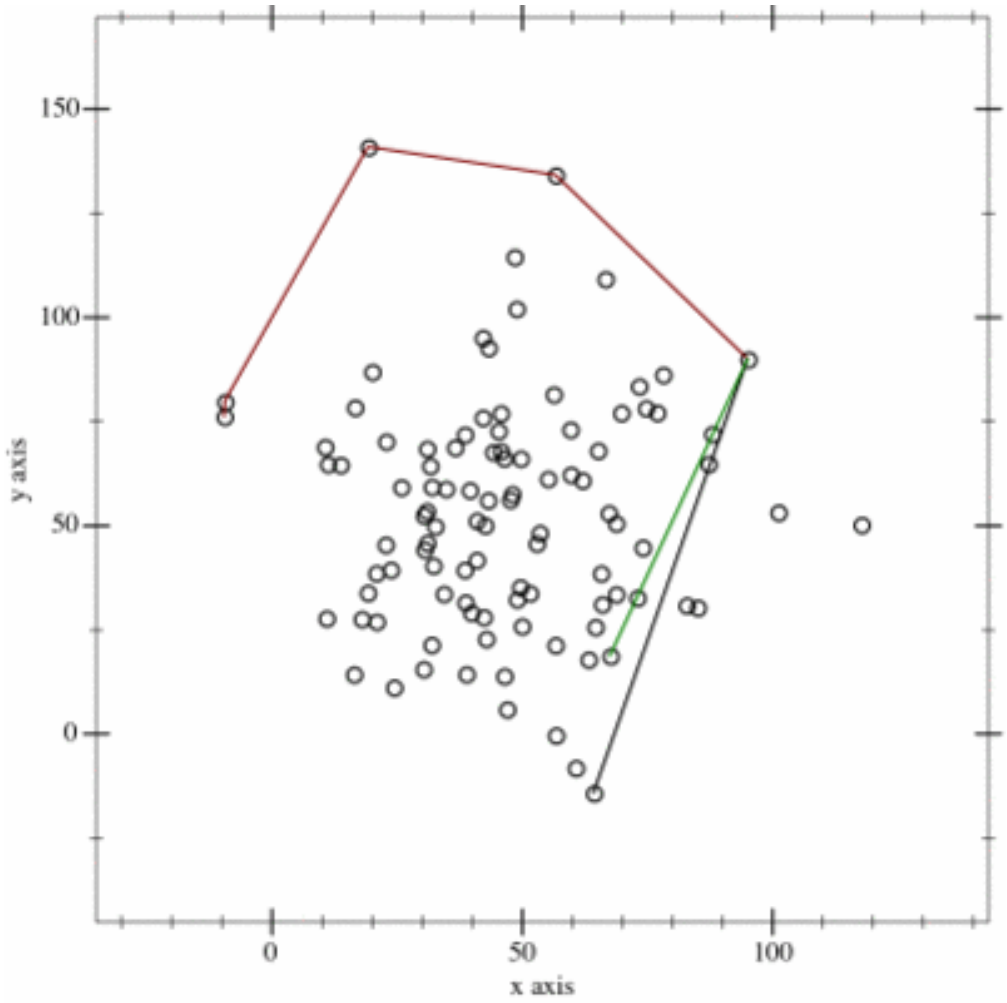


- Skup svih konveksnih kombinacija datog skupa tačaka
- Najmanji konveksni poligon koji sadrži sve tačke iz skupa
- Primene: klasifikacija oblika, detekcija kolizija, istraživanje mešavina . . .

Algoritam 1: Jarvis March (Gift Wrapping)

1. Lociramo najlevlju tačku i dodamo je u rešenje
2. U svakom koraku dodajemo tačku takvu da su sve preostale *desno*
3. Ponavljamo dok ne izgradimo ceo omotač

Algoritam 1: Jarvis March - animacija



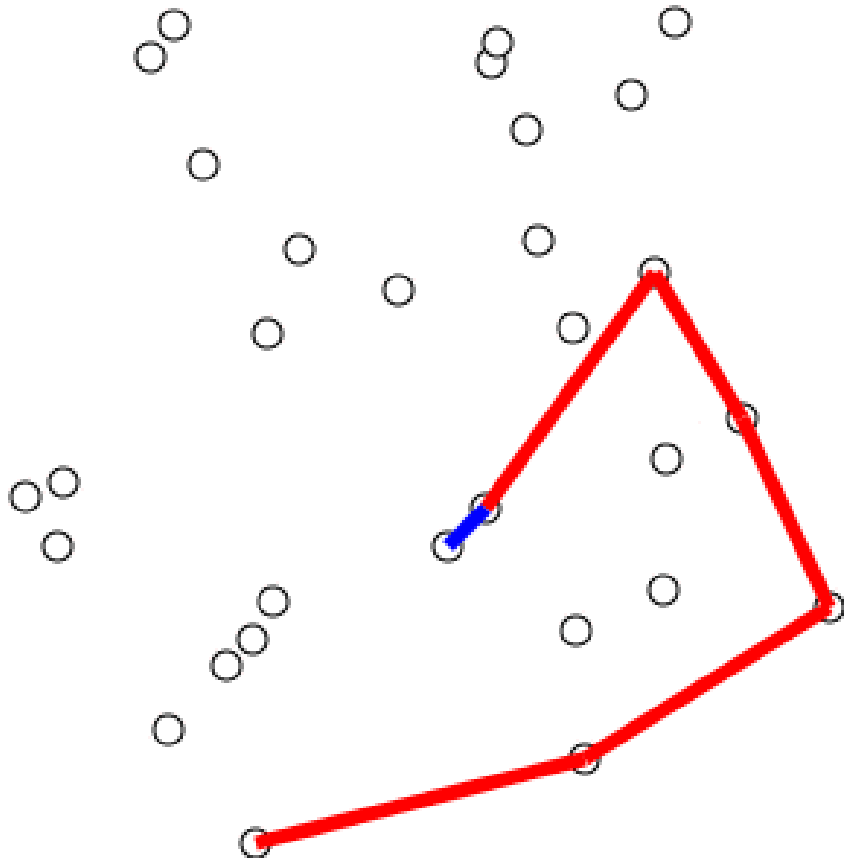
Algoritam 1: Jarvis March (Gift Wrapping)

- $\mathcal{O}(nh)$ (n-broj tačaka u skupu, h-broj tačaka na konveksnom omotaču)
- *Output-sensitive* algoritam
- Jednostavan, ne previše efikasan (osim kada je h jako malo!)

Algoritam 2: Graham Scan

1. Lociramo najdonju-najlevlju tačku, označimo je sa P
2. Koristimo *stek* da čuvamo trenutno izgrađen deo omotača
3. Procesiramo tačke sortirane po uglu koji grade sa P u odnosu na x-osu
4. Kada dodajemo novu tačku uklanjamo tačku sa vrha steka dokle god je zaokret *desno*

Algoritam 2: Graham Scan - animacija



Algoritam 2: Graham Scan

- $\mathcal{O}(n \log n)$
- Dominira sortiranje, ako su tačke sortirane $\mathcal{O}(n)$
- Asimptotski optimalan nezavisno od h, najpoznatiji

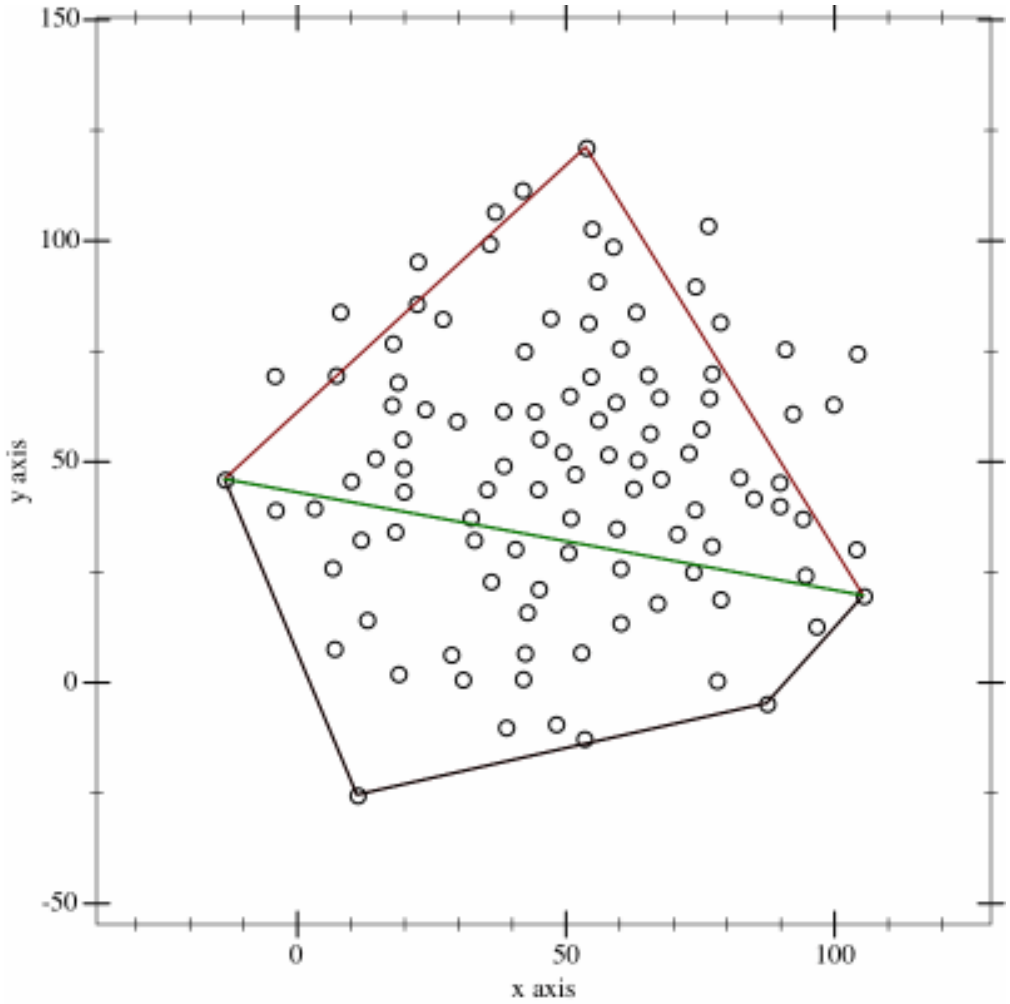
Algoritam 3: Quick Hull

1. Spojimo tačke sa min i max x koordinatom (A i B) i dodamo ih u rešenje

2. Sa obe strane nađemo ekstremnu tačku (C), dodamo je u rešenje, i napravimo trougao ABC

3. Tačke unutar ΔABC odbacimo, rekurzivno ponavljamo za dva dobijena podskupa

Algoritam 3: Quick Hull - animacija



Algoritam 3: Quick Hull

- *Worst-case* složenost: $\mathcal{O}(n^2)$
- *Best-case* složenost: $\mathcal{O}(n \log n)$
- Brz u praksi, lako uopštiv na više dimenzija

Algoritam 4: Kirkpatric-Seidel (*Ultimate Algorithm*)

1. Tražimo samo gornji deo konveksnog omotača (analogno za donji)
2. Nađemo medijanu x-koordinata tačaka (x_{med})
3. U $\mathcal{O}(n)$ nađemo *most* - stranicu omotača koja seče pravu $x = x_{med}$
4. Rekurzivno obrađujemo dva skupa tačaka sa leve tj. desne strane prave $x = x_{med}$
5. Usput odbacujemo veliki broj tačaka optimizacijama

Algoritam 4: Kirkpatric-Seidel - interaktivni demo

http://ben-tanen.com/ultimate-convex-hull/

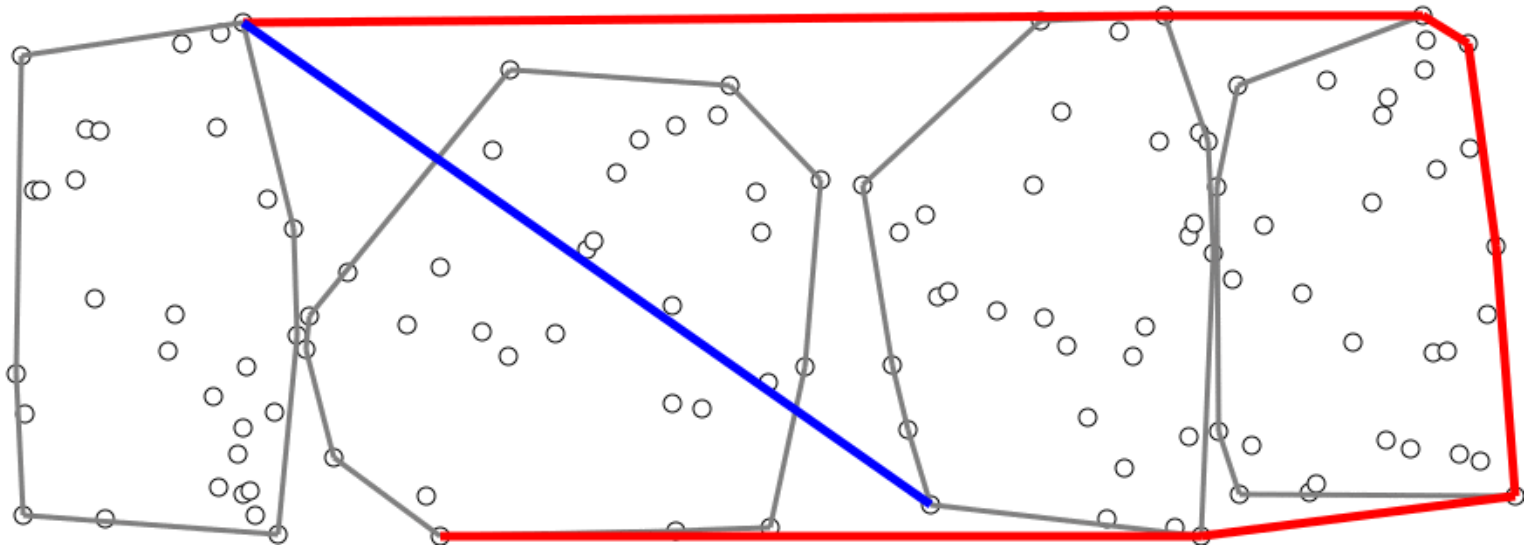
Algoritam 4: Kirkpatric-Seidel (*Ultimate Algorithm*)

- $\mathcal{O}(n \log h)$
- Najstariji optimalan *output-sensitive* algoritam
- Spor u praksi

Algoritam 5: Chan's Algorithm

1. Biramo neko m (i nadamo se da je $m \geq h$)
2. Nasumično podelimo tačke u grupe veličine m
3. Za svaku od n/m grupa nađemo *mini-omotač Graham Scan* algoritmom
4. Izvršavamo *Gift Wrapping* ali razmatramo samo tangente na *mini-omotače*
5. Stajemo nakon m koraka ako nismo našli ceo omotač, i biramo veće m dok ga ne nađemo

Algoritam 5: Chan's Algorithm - animacija



- Interaktivni demo: http://chrisgregory.me/projects/web/chans/

Algoritam 5: Chan's Algorithm

- Jedna iteracija: $\mathcal{O}(n \log m)$ + $\mathcal{O}(h\frac{n}{m} \log m)$
- Ovo je $\mathcal{O}(n \log h)$ za $m \sim h$
- Ako biramo m kao $2^{2^{iteracija}}$ ukupna složenost ostaje $\mathcal{O}(n \log h)$
- Iste složenosti ali jednostavniji od Kirkpatric-Seidel
- Lako uopštiv na više dimenzija

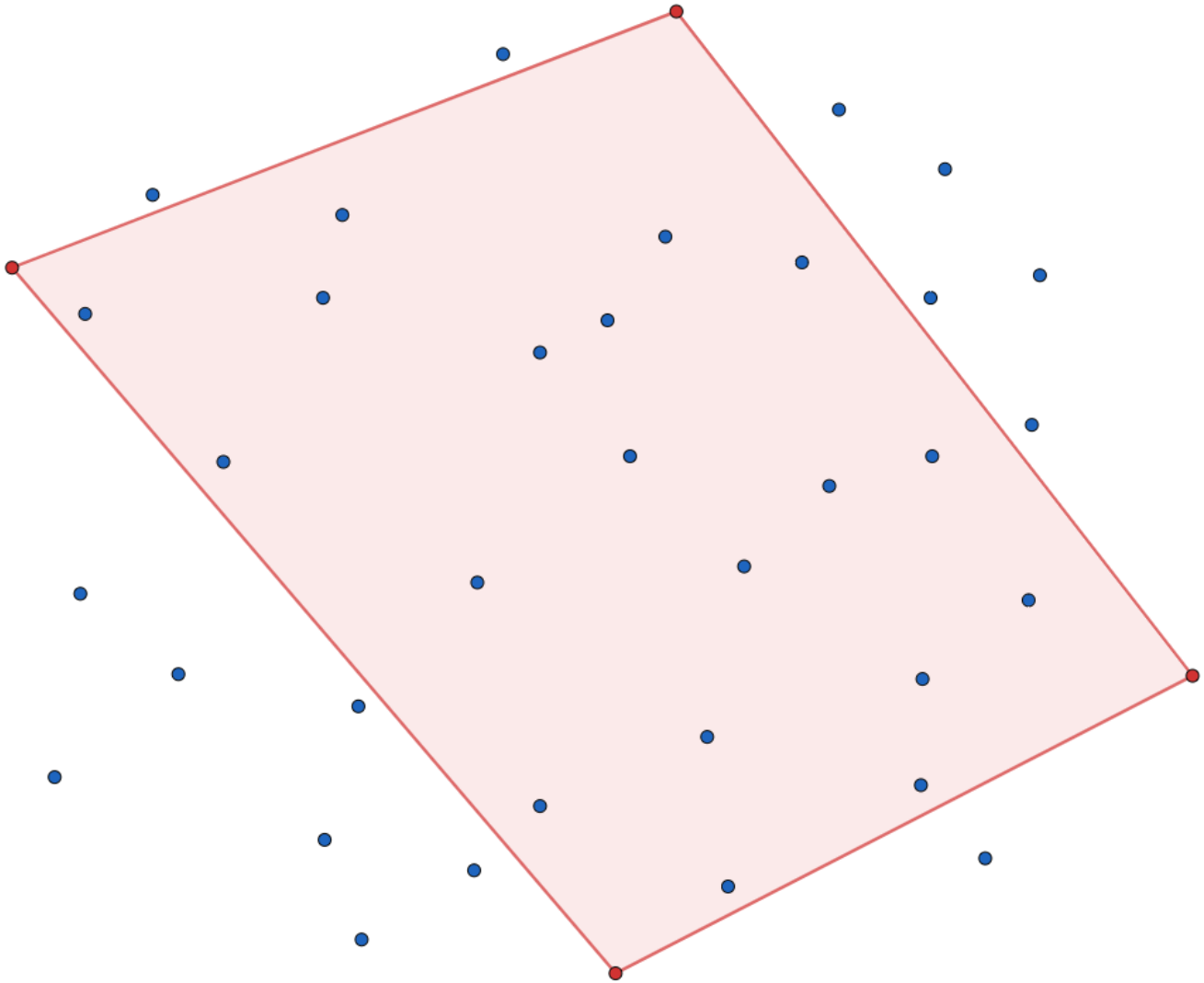
tl;dr

1. *Jarvis March*: $\mathcal{O}(nh)$, najjednostavniji
2. *Graham Scan*: $\mathcal{O}(n \log n)$, optimalan a da nije *output-sensitive*
3. *Quick Hull*: $\mathcal{O}(n \log n)$ u najboljem, $\mathcal{O}(n^2)$ u najgorem, brz u praksi
4. *Kirkpatric-Seidel (Ultimate Algorithm)*: $\mathcal{O}(n \log h)$, optimalan *output-sensitive*
5. *Chan's Algorithm*: $\mathcal{O}(n \log h)$, dosta jednostavniji

Bonus: Akl-Toussaint heuristika

- Često se koristi kao prvi korak nekog od ovih algoritama
- Nađemo 4 tačke sa ekstremnim x i y koordinatama, zatim uklanjamo sve tačke u tom četvorouglu jer ne mogu biti deo konveksnog omotača
- $\mathcal{O}(n)$
- Varijacija: 8 tačaka (ekstremno $x + y$, itd.)

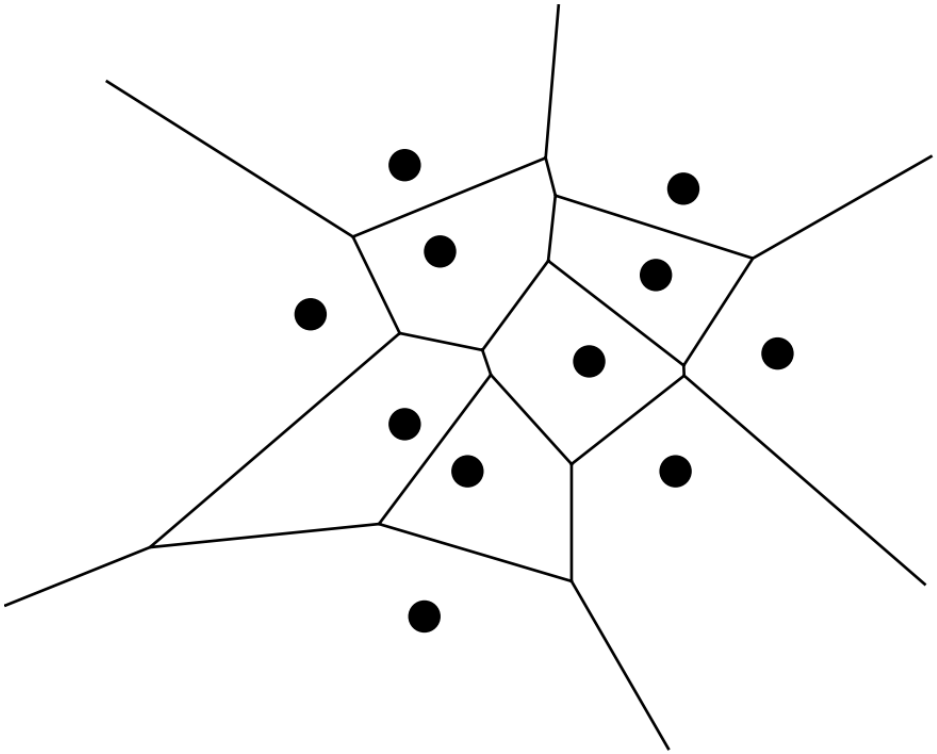
Bonus: Akl-Toussaint heuristika



Voronoi dijagram

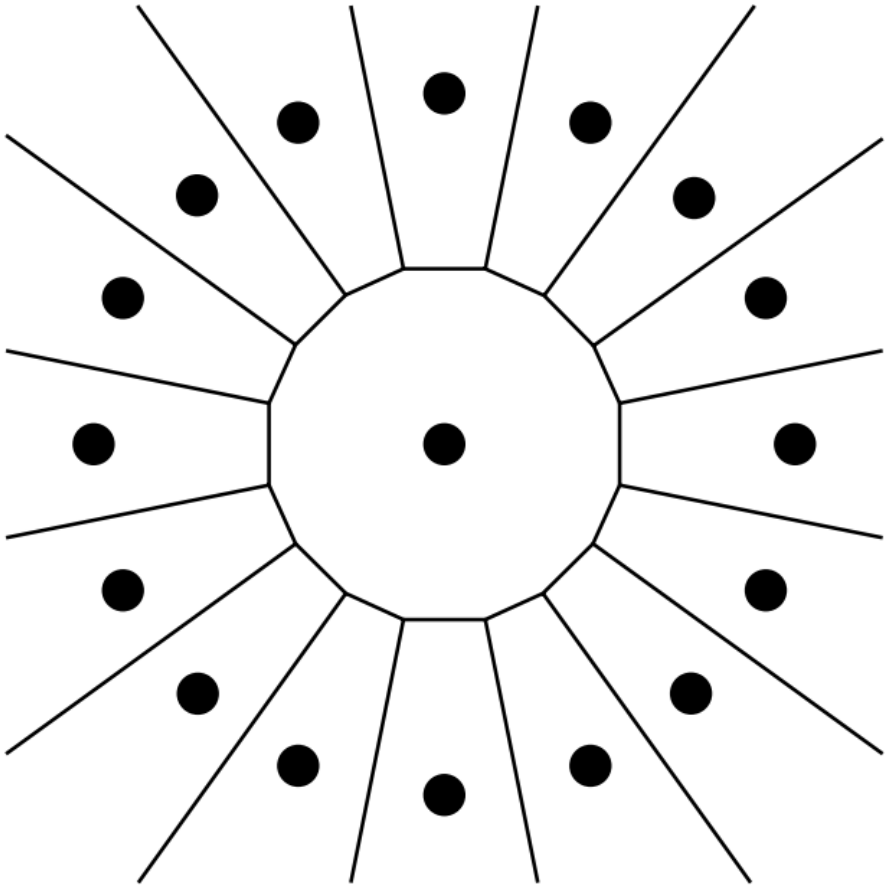
Motivacija: najbliži vatrogasac - animacija

Definicija

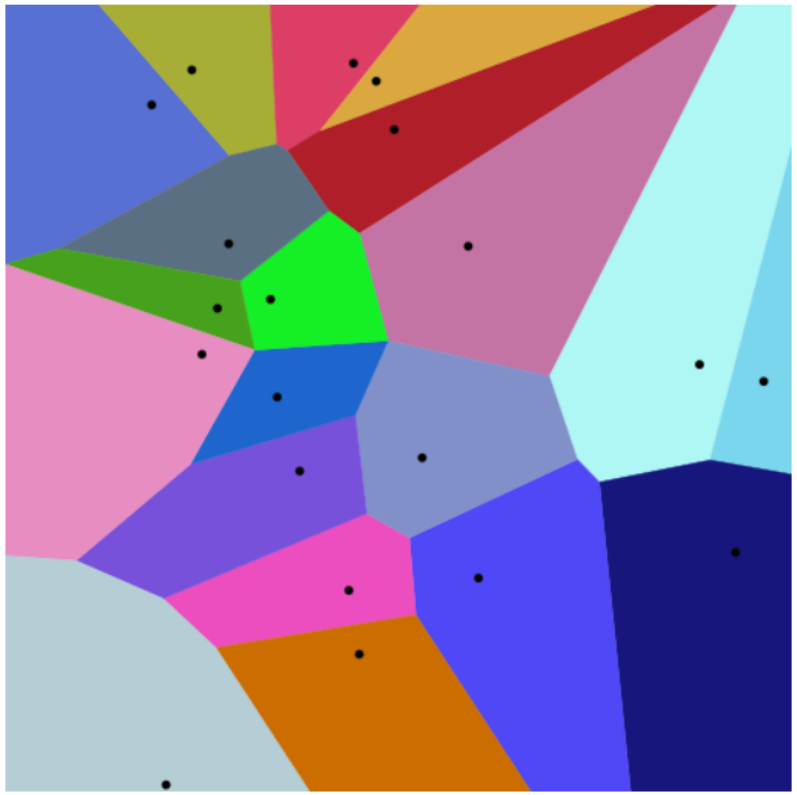


- Podela ravni na ćelije u odnosu na datih N tačaka
- Ćelija tačke A: skup tačaka kojima je A najbliža

Definicija



Definicija

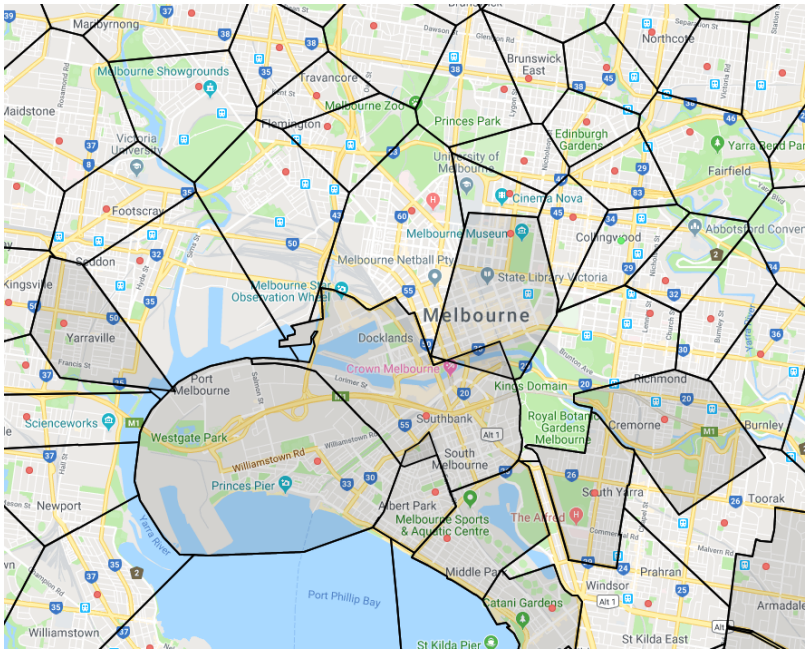


- Čvorovi dijagrama jednako udaljeni od 3 tačke (centri opisanih krugova)
- Grane dijagrama jednako udaljene od 2 tačke (simetrale duži)

Interaktivni demo

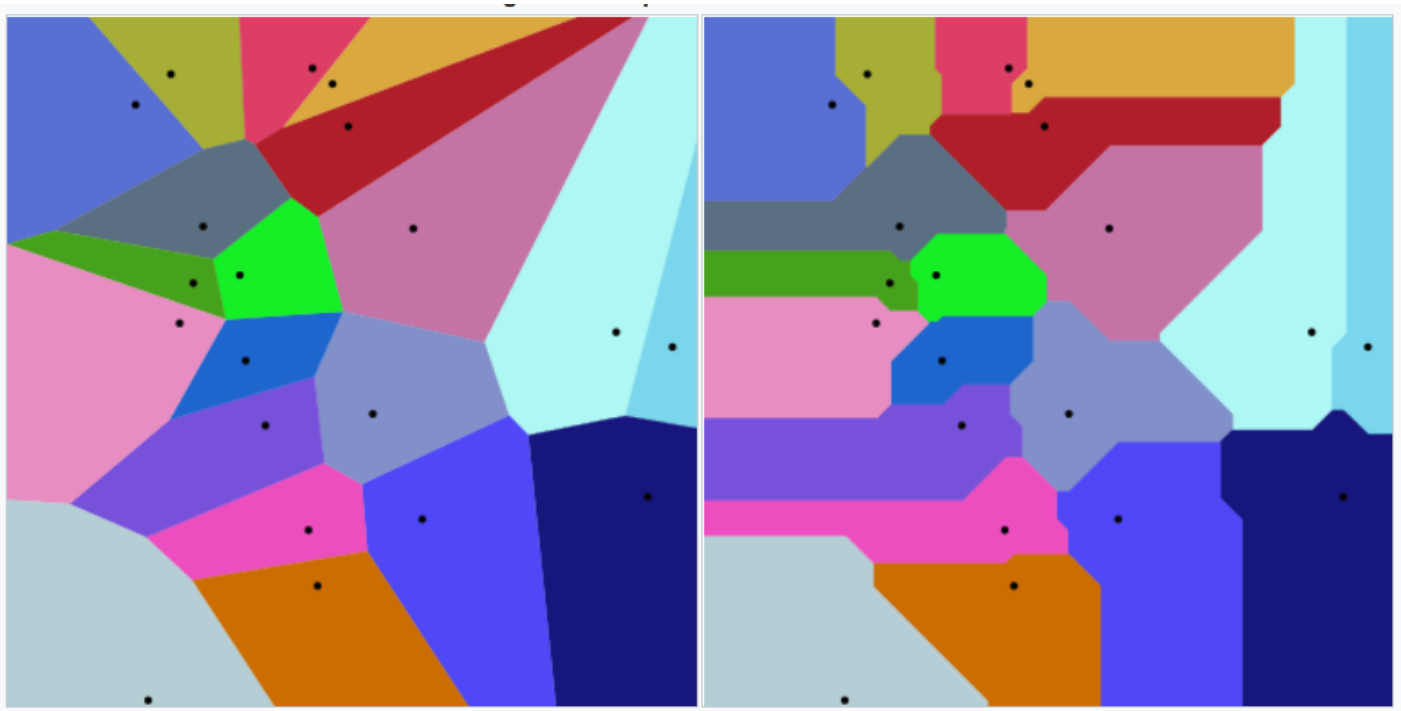
http://alexbeutel.com/webgl/voronoi.html

Primena



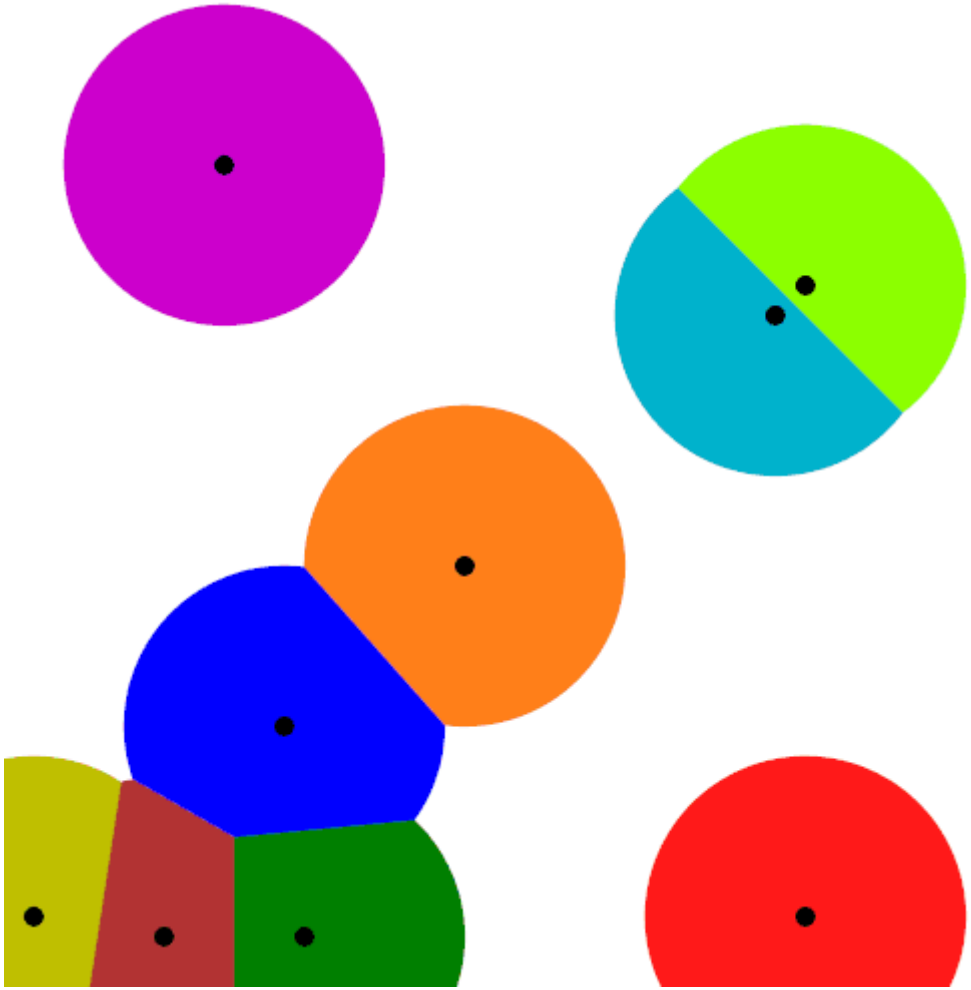
- Grafika, AI, mašinsko učenje, računarske mreže . . .
- Biologija, ekologija, astrofizika, medicina, avijacija, arhitektura . . .

Varijacije



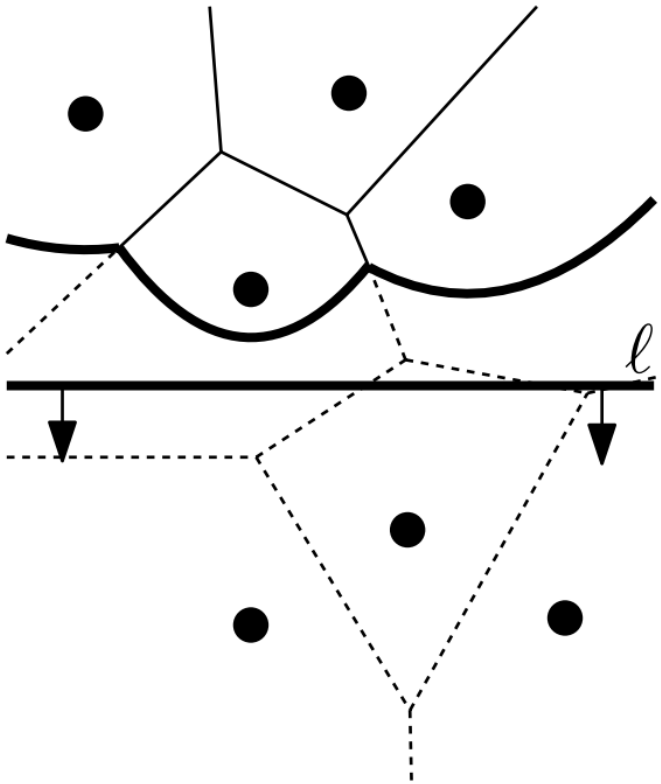
- Menhetn udaljenost, dijagram n-tog reda, težinski, dijagram najudaljenije tačke, dijagram za duži . . .

Algoritmi - floodfill animacija



- Razni; najkorišćeniji: **Fortune's algoritam**

Fortune's algoritam



- *Sweep line* ka dole, ocrtavamo dijagram usput

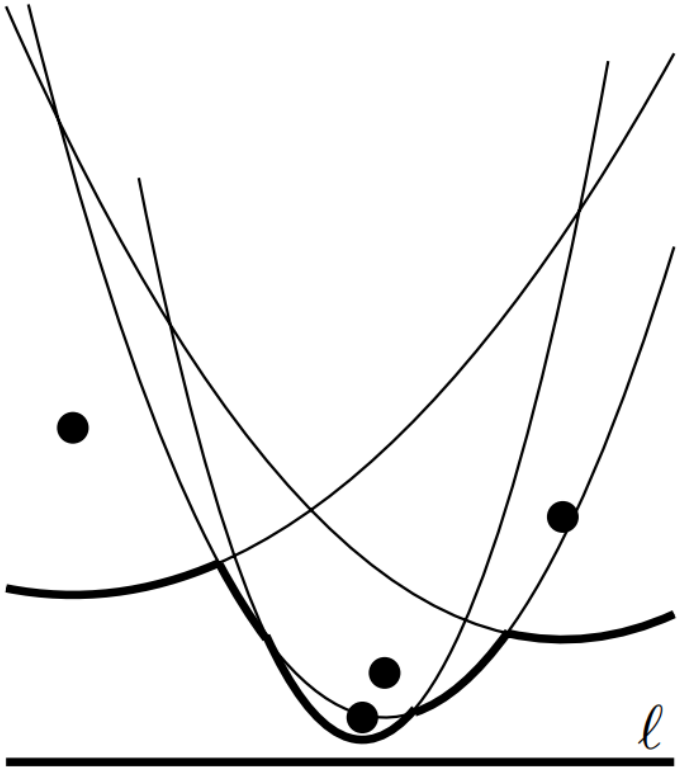
Fortune's algoritam

- Svaka tačka pamti skup tačaka bližih njoj nego *sweep* liniji
- Kako?

Fortune's algoritam

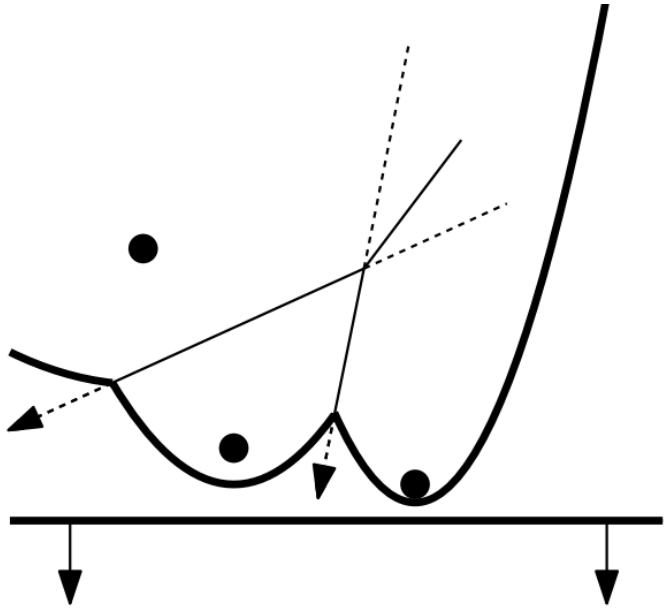
- Svaka tačka pamti skup tačaka bližih njoj nego *sweep* liniji
- Kako?
- Parabole!

Fortune's algoritam: *beach line*



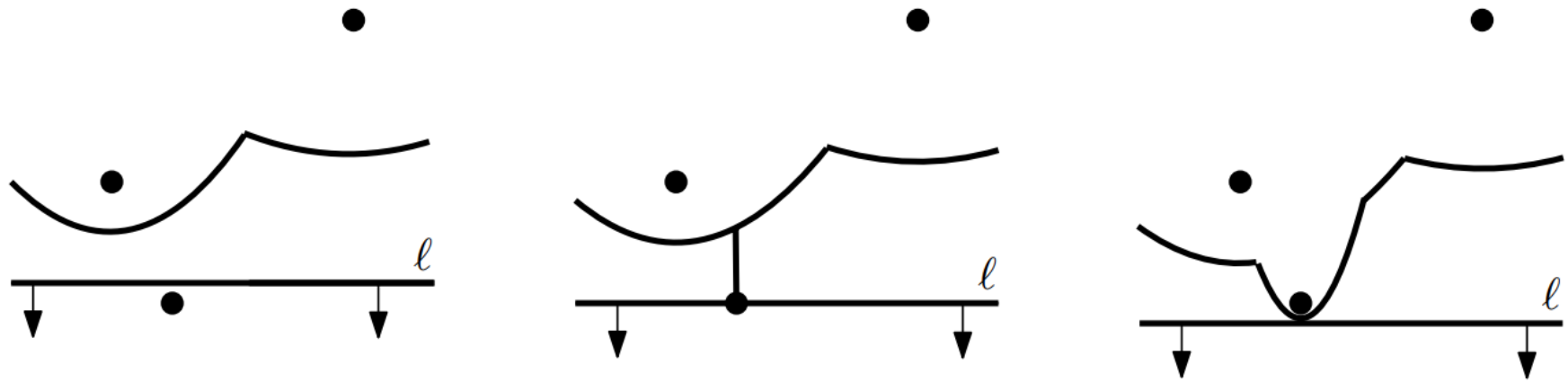
- *Beach line*: skup najnižih delova parabola, iznad njega imamo gotov dijagram

Fortune's algoritam: *beach line*



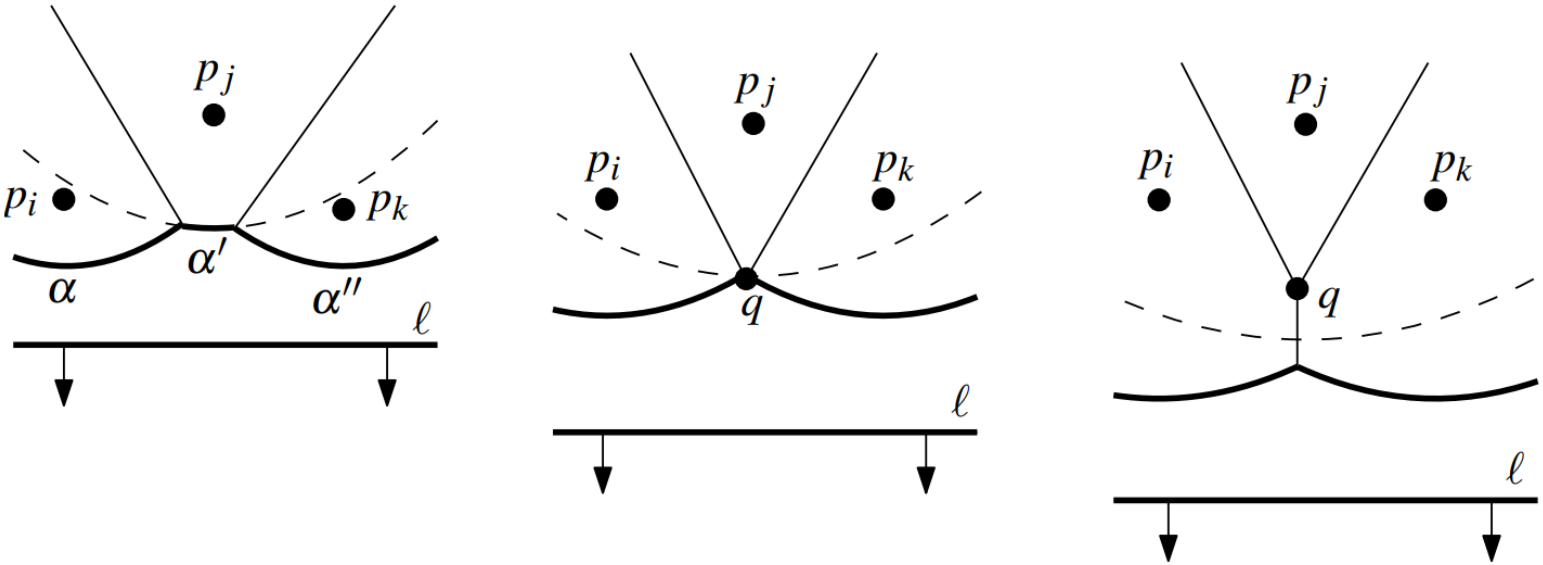
- Prelomne tačke ocrtavaju grane Voronoi dijagrama
- Dva tipa događaja: *site event* i *circle event*

Fortune's algoritam: *site event*



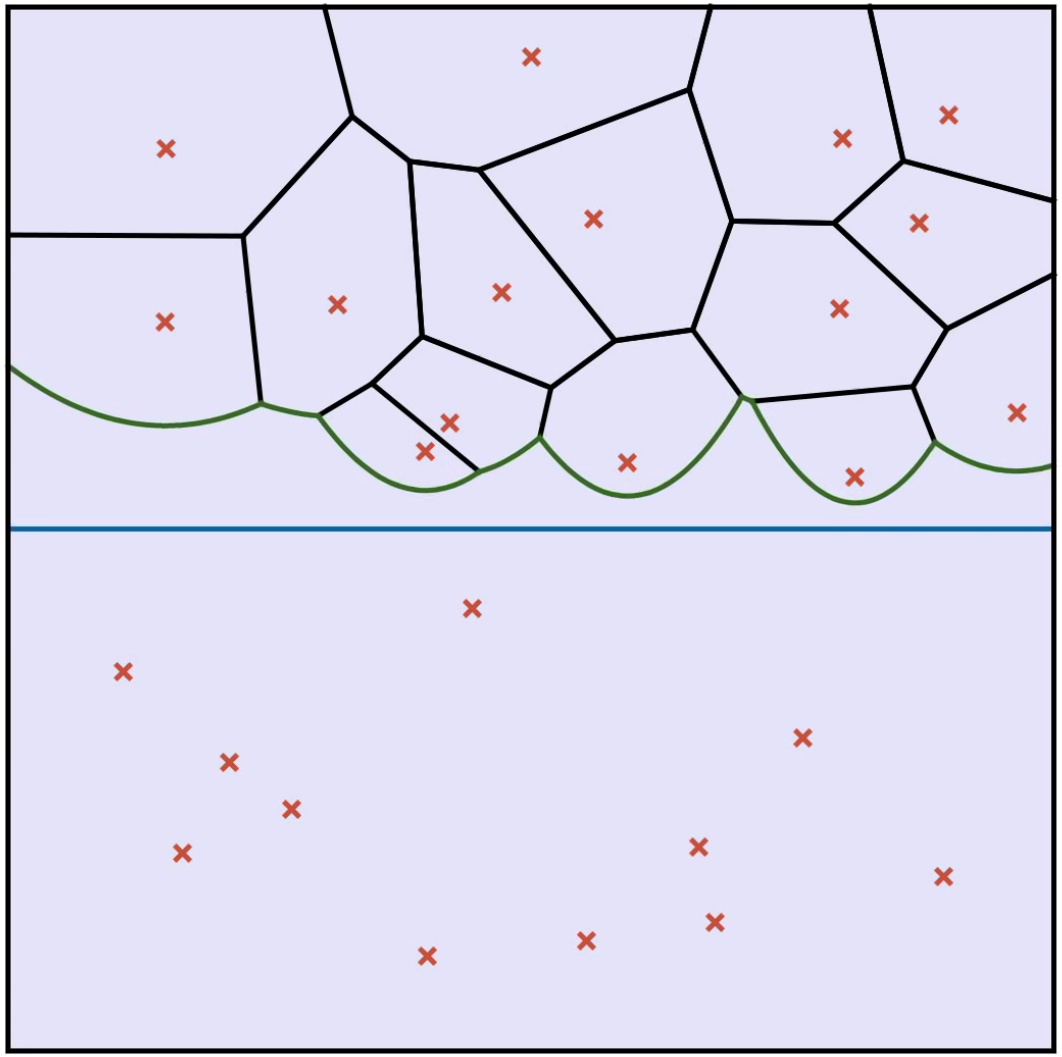
- Dešava se kada *sweep* linija udari u tačku
- Dodaje se novi luk na *beach line*

Fortune's algoritam: *circle event*

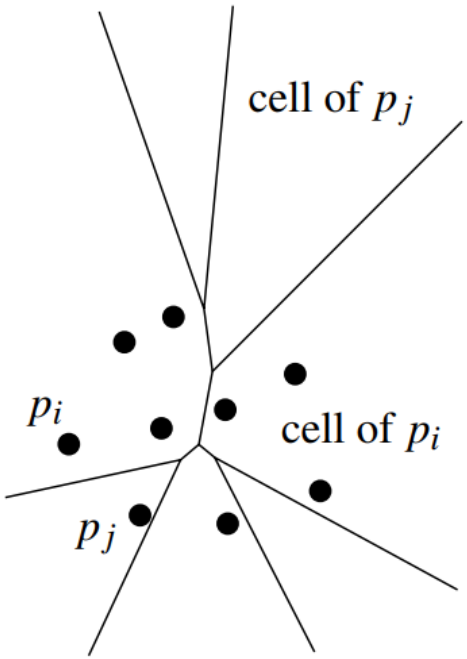


- Dešava se kada *sweep* linija udari u dno kruga nad tri tačke
- Nestaje luk sa *beach line* i nastaje teme dijagrama

Fortune's algoritam - animacija

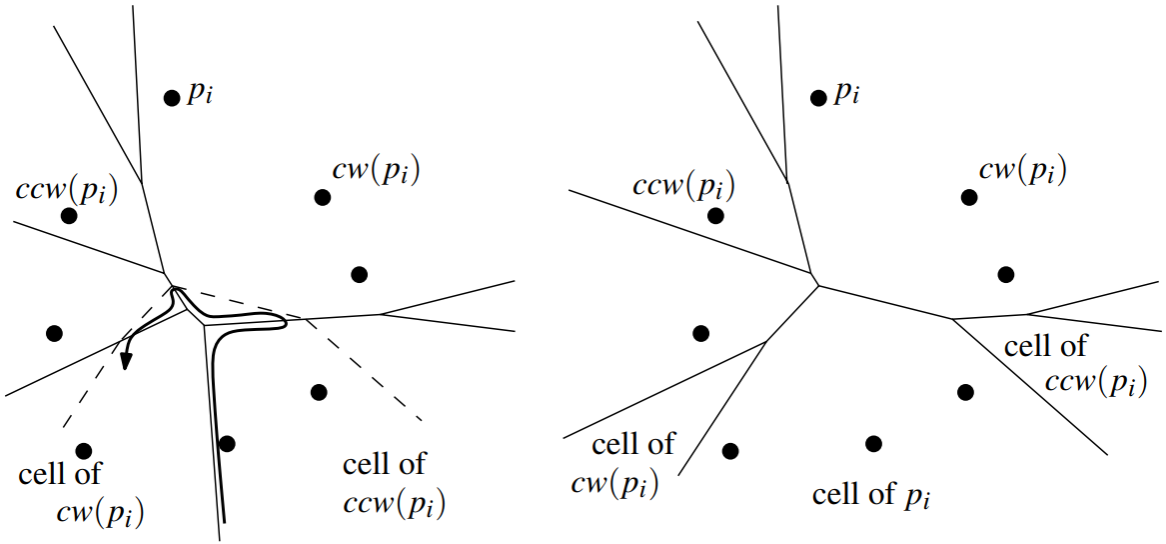


Voronoi dijagram najudaljenije tačke



- Ćelija tačke A: skup tačaka kojima je A **najdalja**

Voronoi dijagram najudaljenije tačke

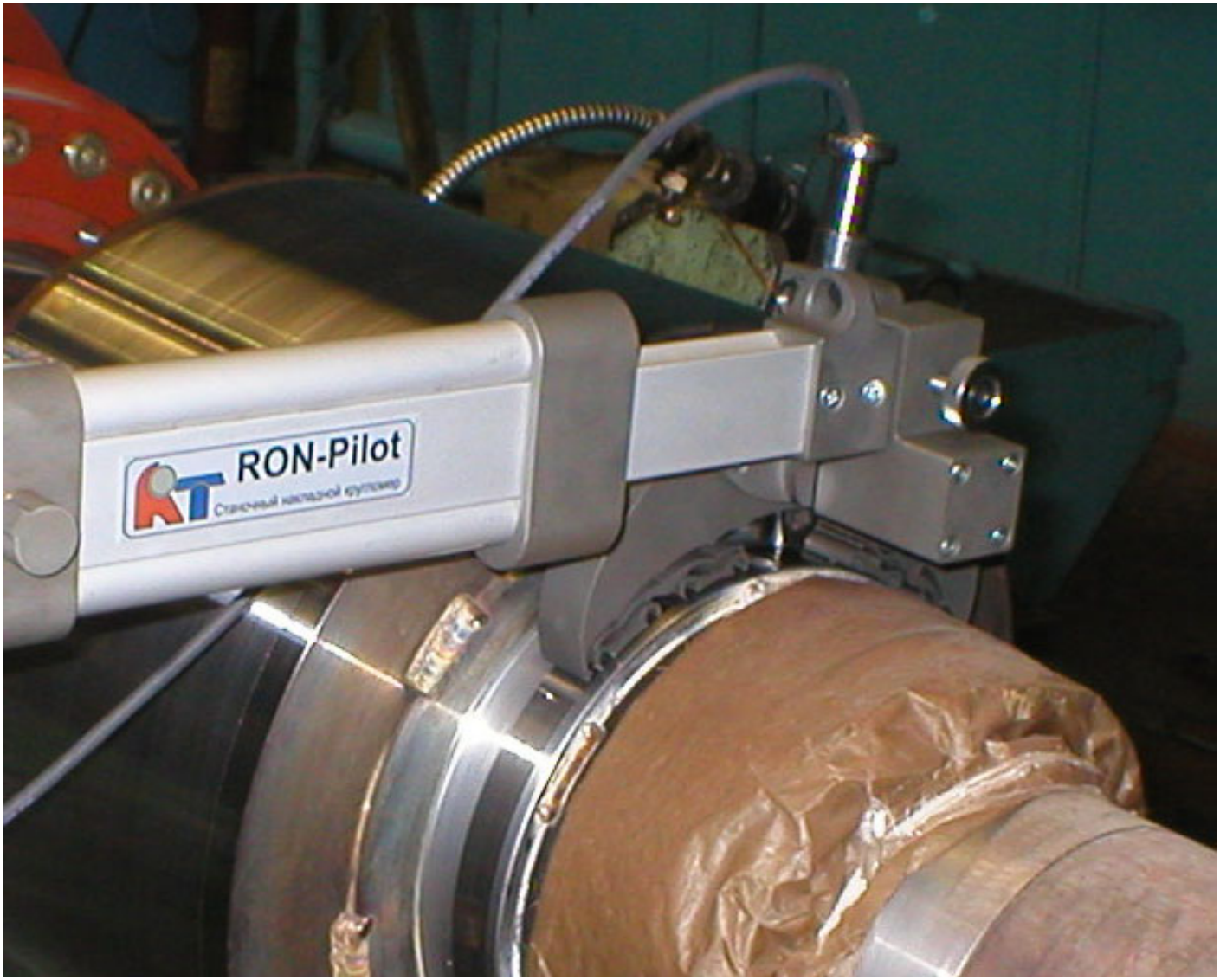


- Inkrementalni algoritam: krećemo sa 3 tačke i dodajemo jednu po jednu
- Usput modifikujemo trenutno stanje dijagrama
- $\mathcal{O}(n \log n)$

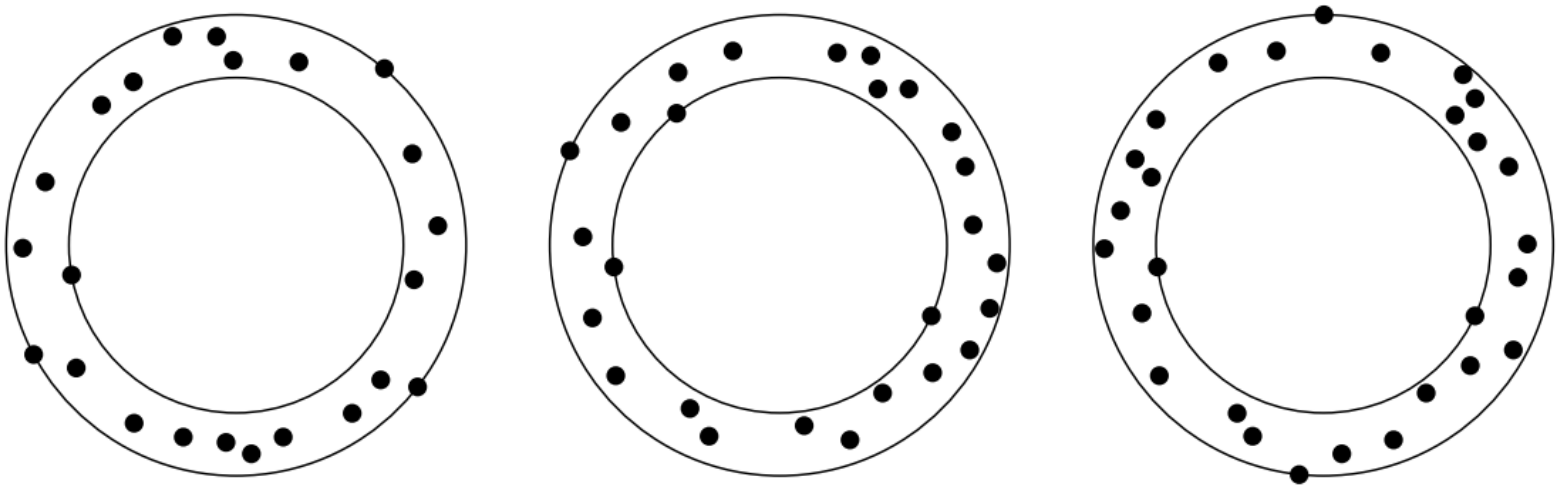
tl;dr

- Voronoi dijagram je podela ravni na ćelije koje označavaju najbližu tačku
- Ima velike primene u raznim oblastima kao i varijacije
- Računa se Fortune's algoritmom u $\mathcal{O}(n \log n)$

Bonus: *roundness*

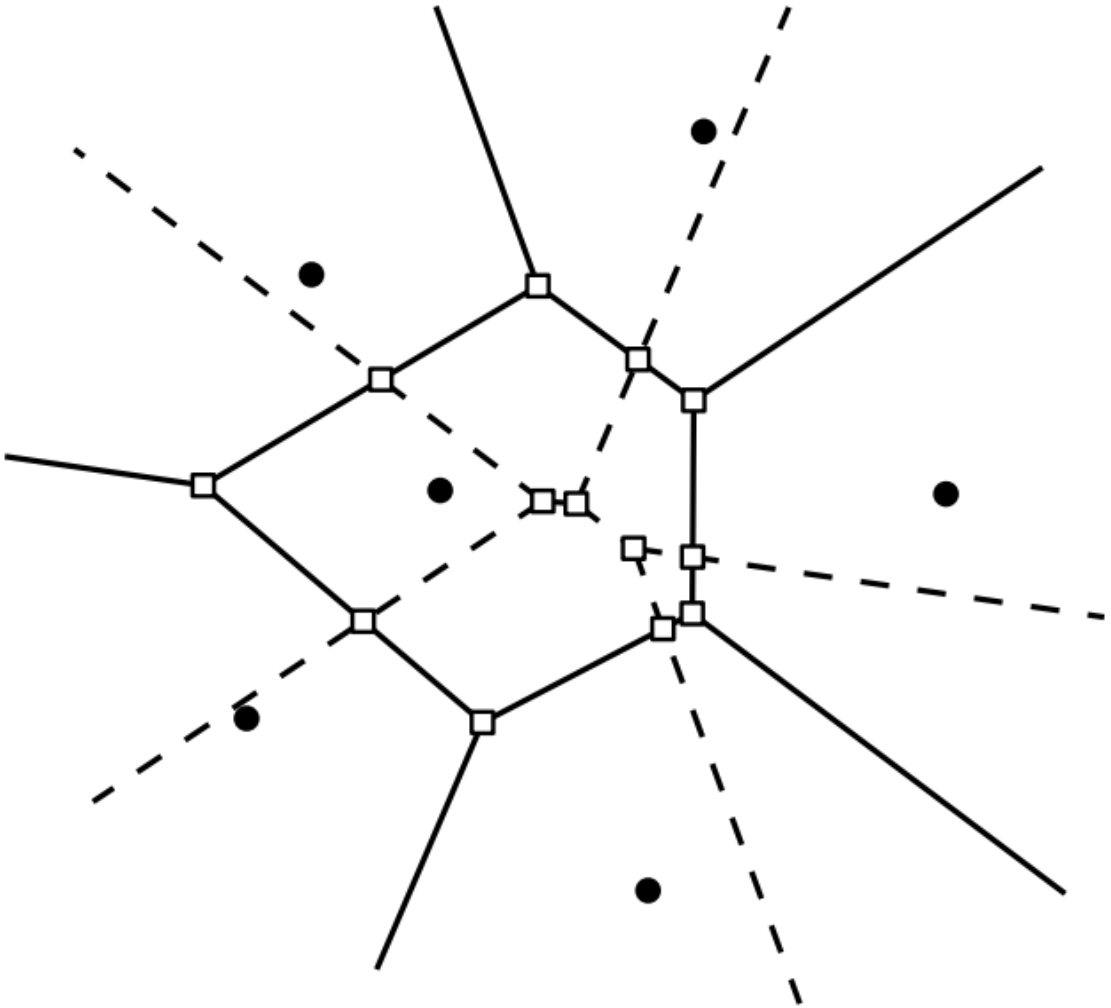


Minimalni kružni prsten



- Treba naći najuži kružni prsten (K_{out}, K_{in}) oko skupa tačaka
- Tri slučaja:
  1. 3 tačke na K_{out} (centar je teme dijagrama najdalje tačke)
  2. 3 tačke na K_{in} (centar je teme Voronoi dijagrama)
  3. Po 2 tačke na oba (centar je presek stranica oba dijagrama)

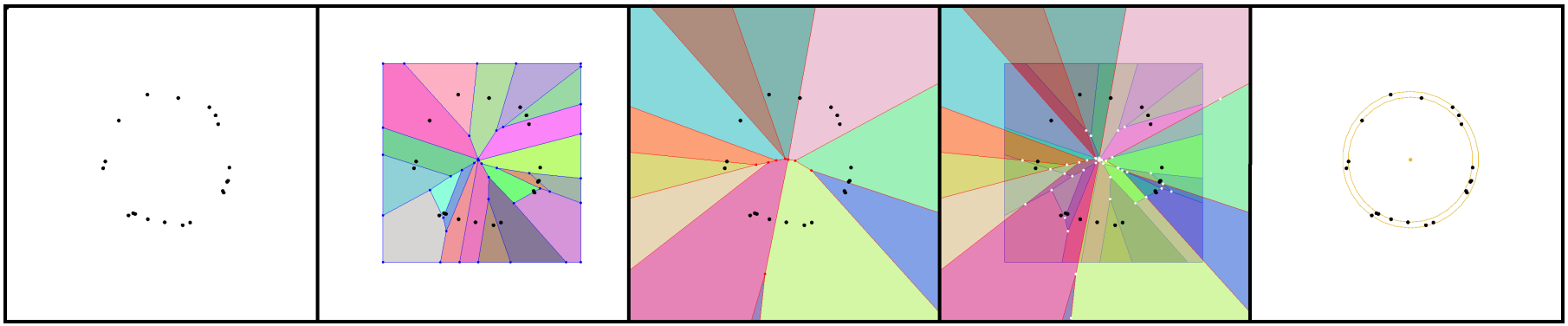
Minimalni kružni prsten



Minimalni kružni prsten

1. Nađemo Voronoi dijagram *Fortune's algoritmom*
2. Nađemo Voronoi dijagram za najudaljenije tačke inkrementalnim algoritmom
3. Presečemo stranice ova dva dijagrama (*višeslojne mape*)
4. Odredimo najmanji kružni prsten uzimajući te preseke i temena oba dijagrama za kandidate (potreban *point location*)

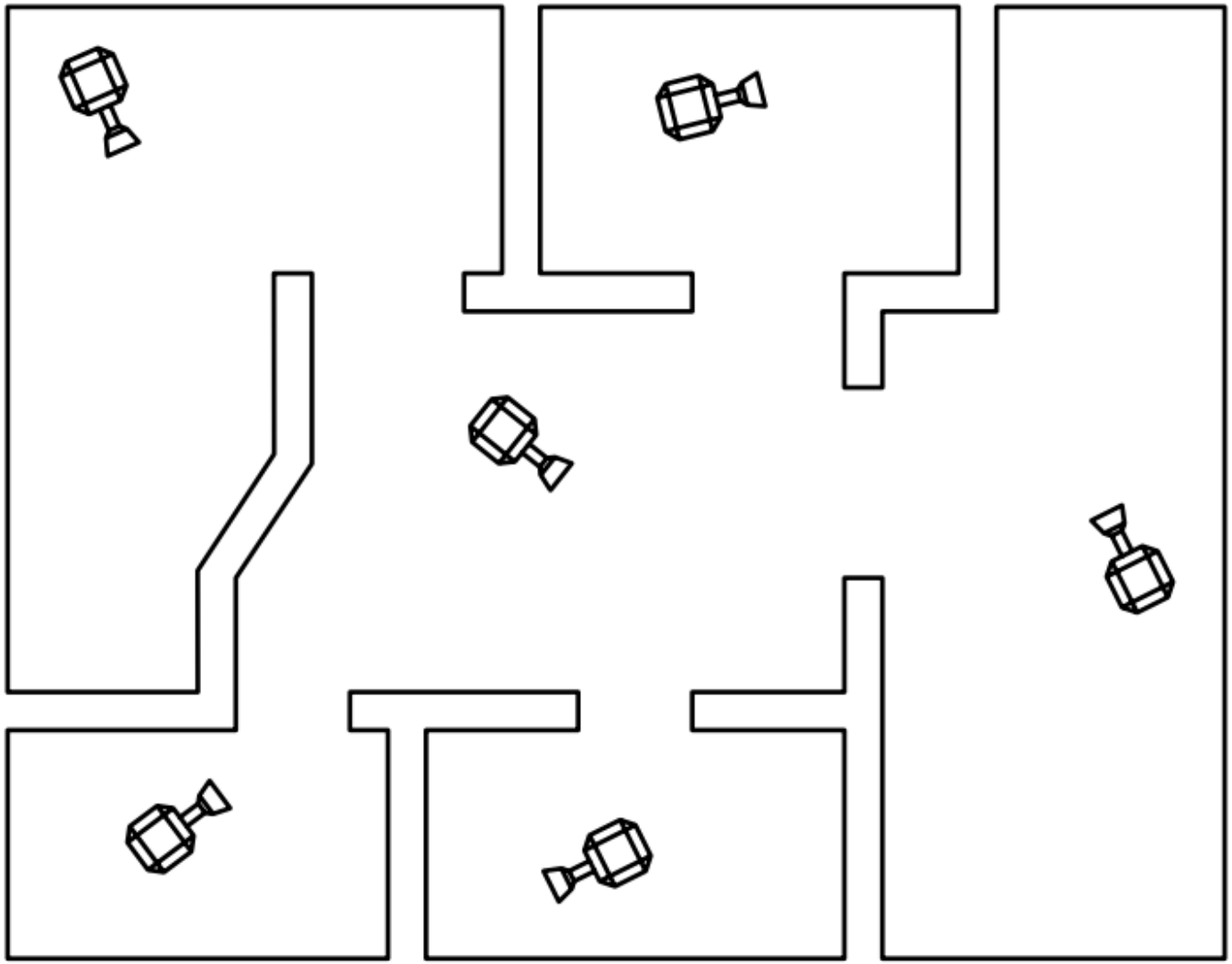
Minimalni kružni prsten



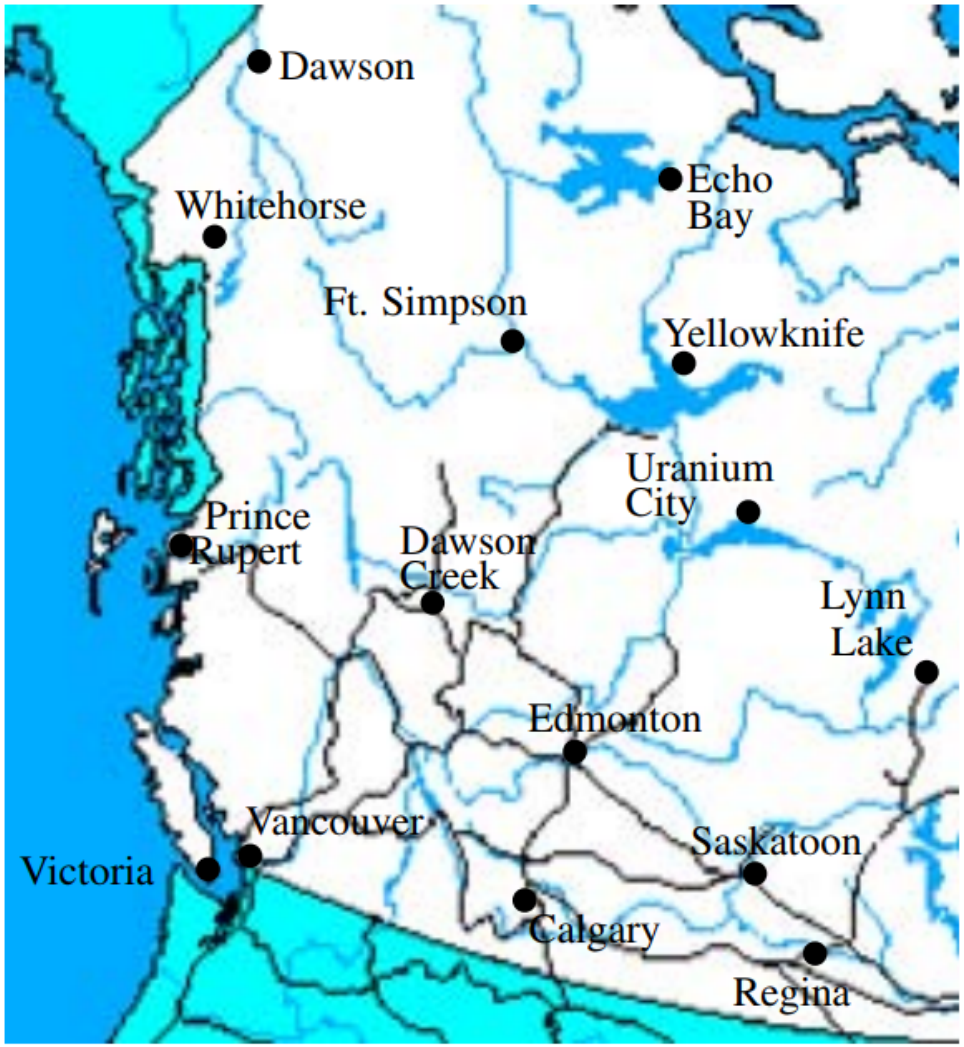
- github.com/rand0musername/min-annulus

Honourable mentions

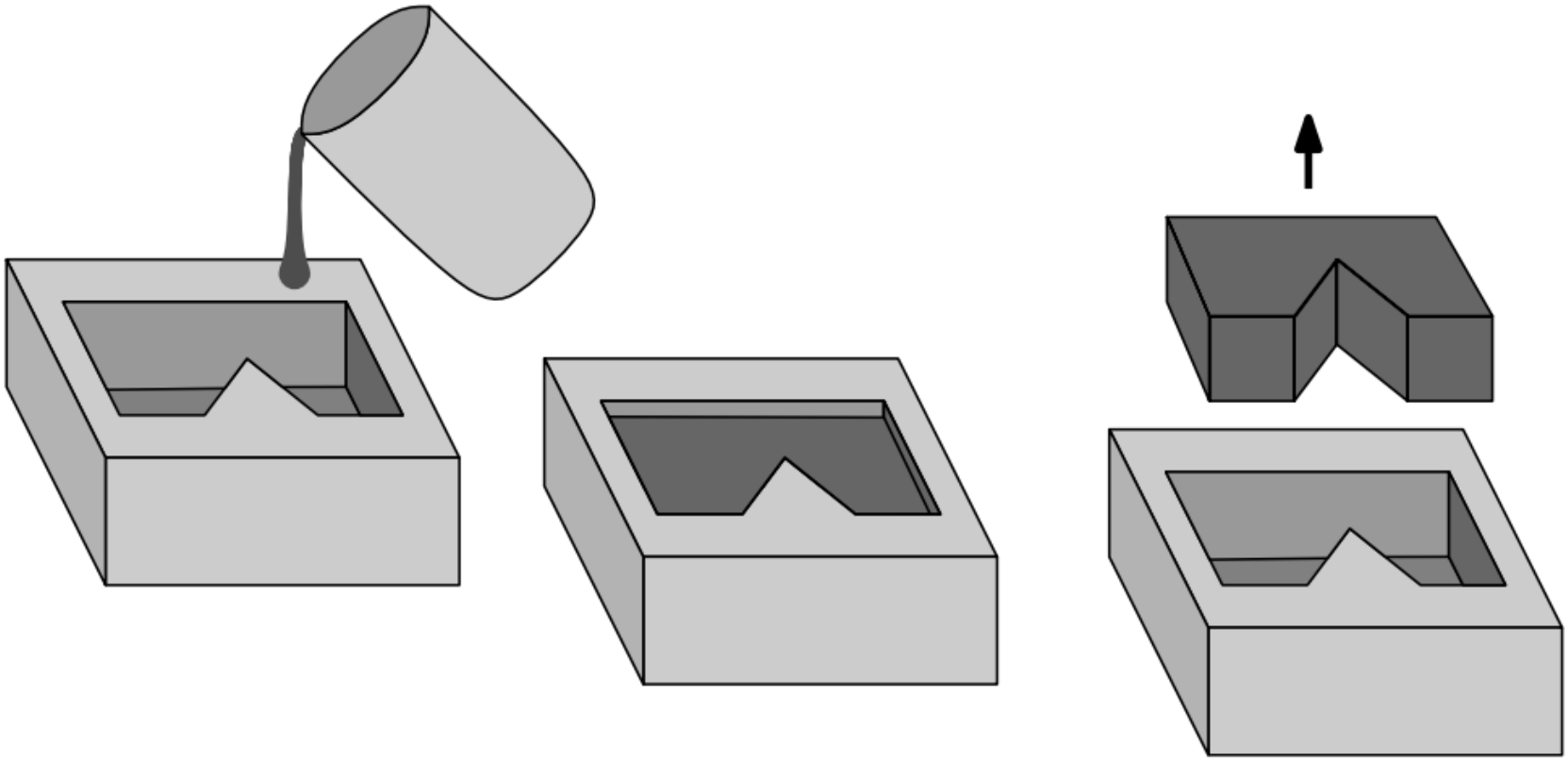
Nadzor galerije: Triangulacija



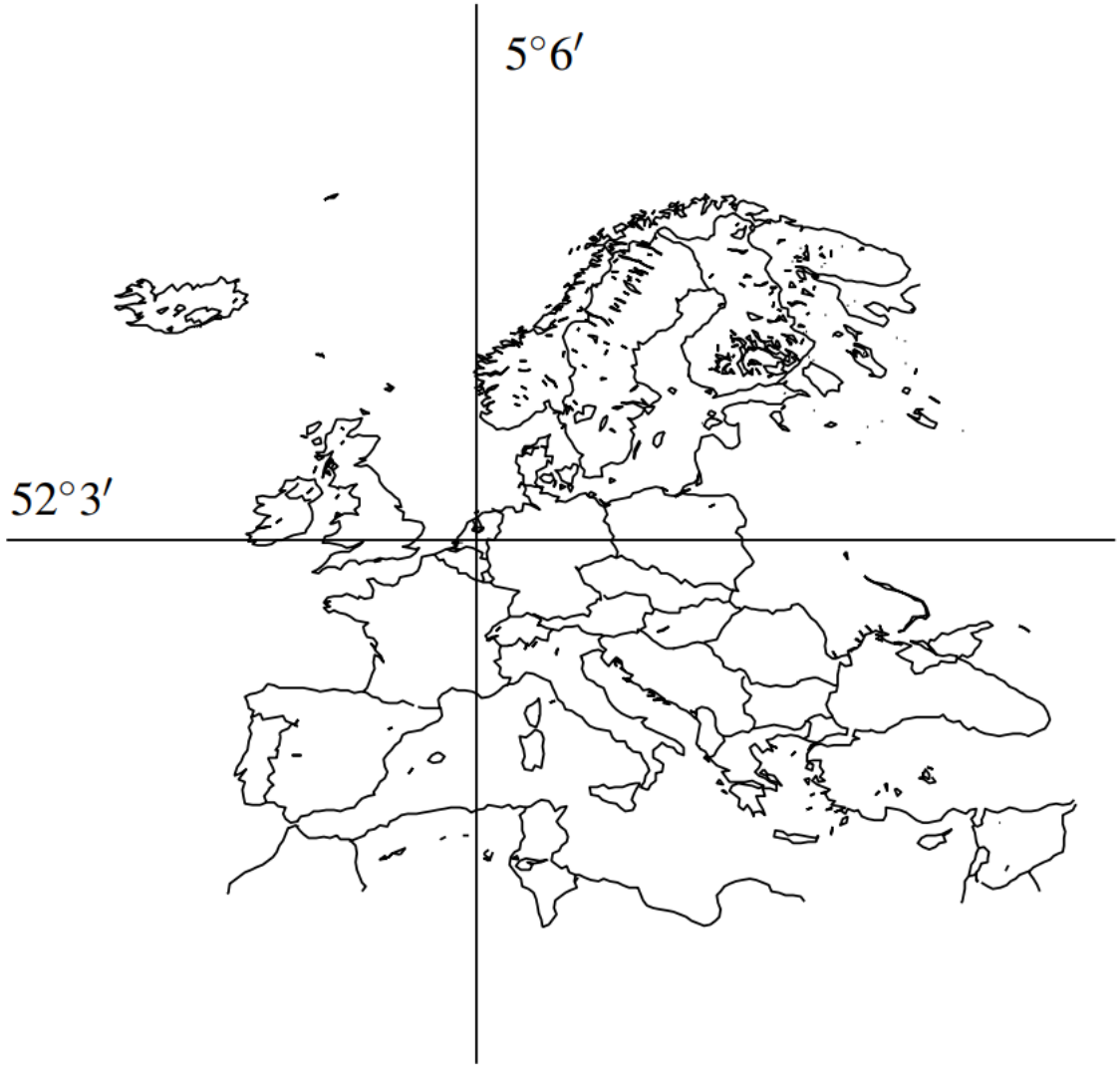
Višeslojne mape: Sweep line



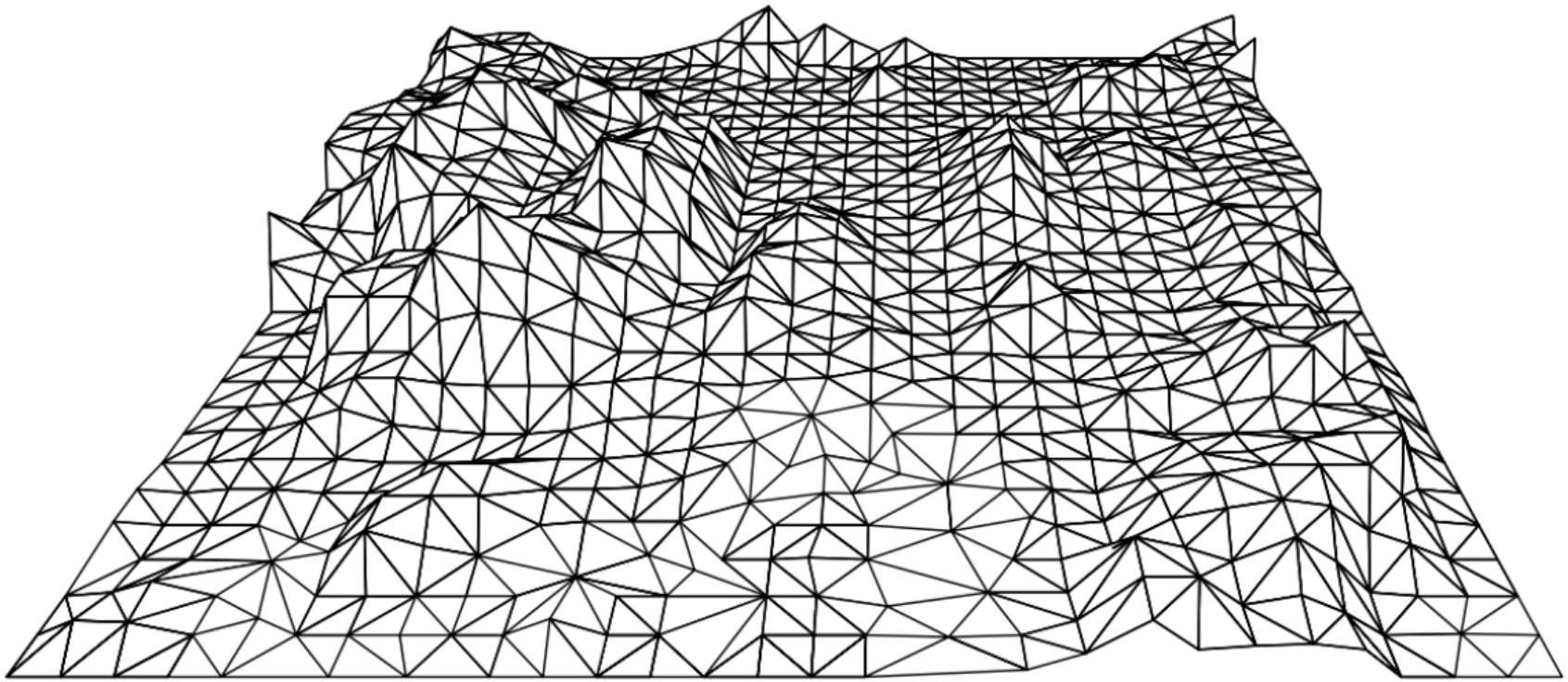
Izlivanje kalupa: Linearno programiranje



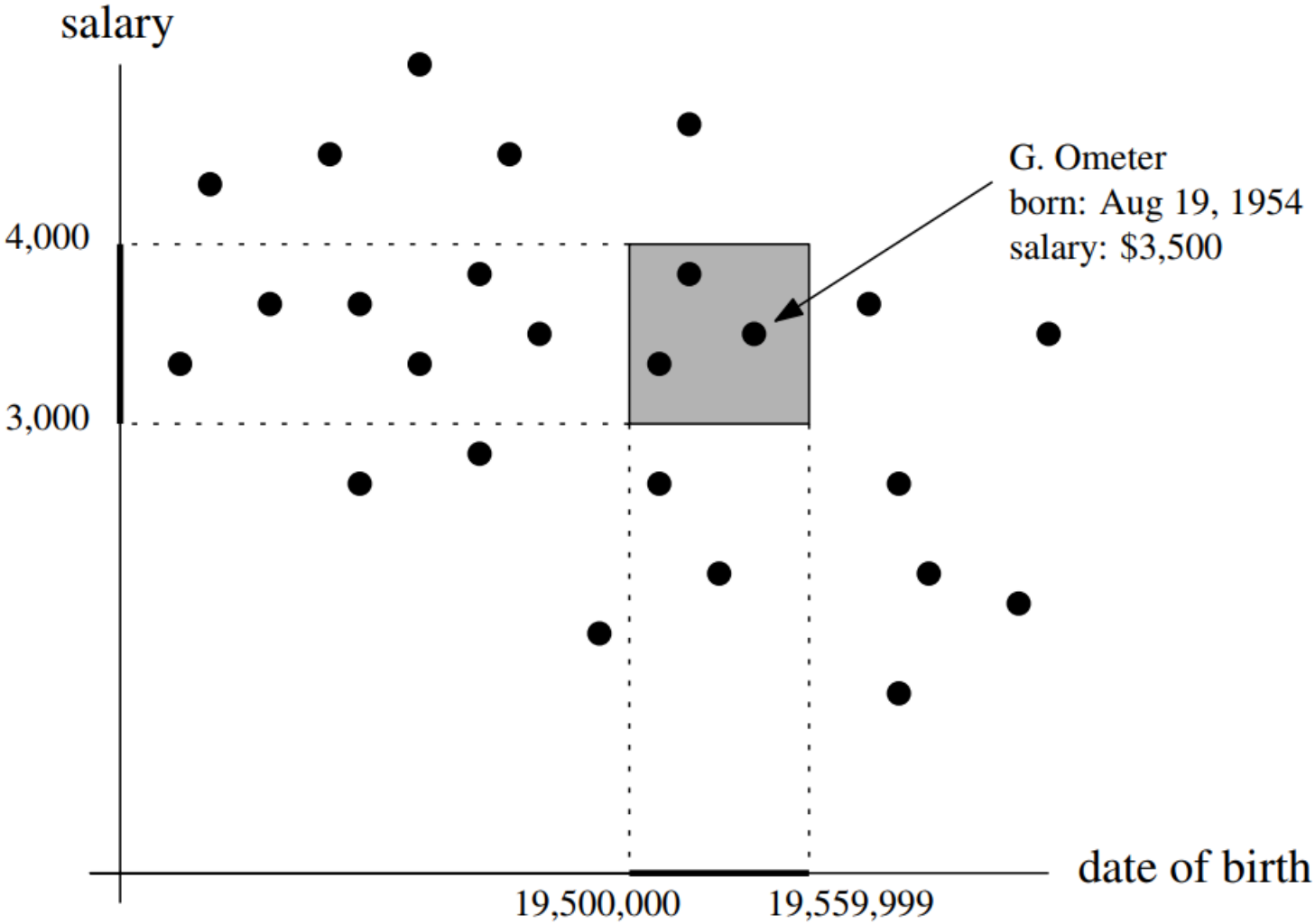
Lociranje tačke: Trapezoidne mape



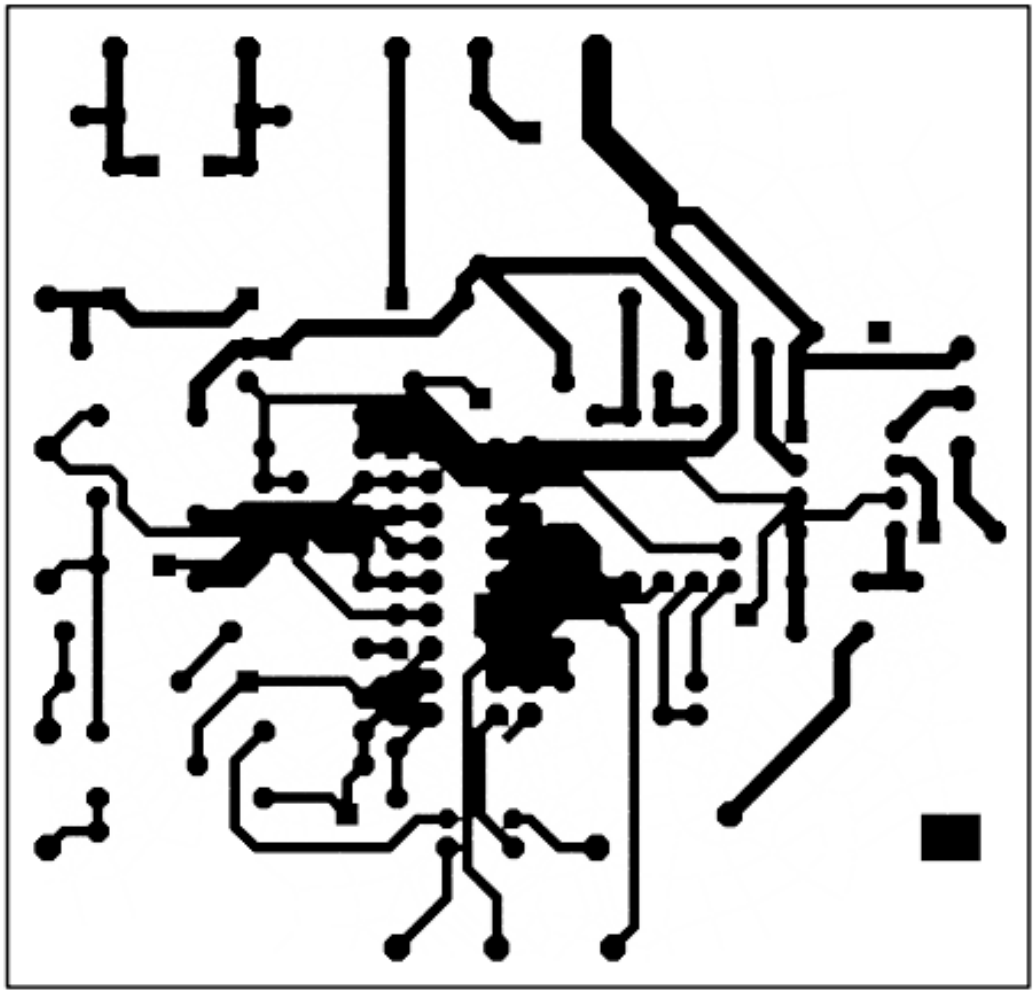
Interpolacija visine: Delonijeva triangulacija



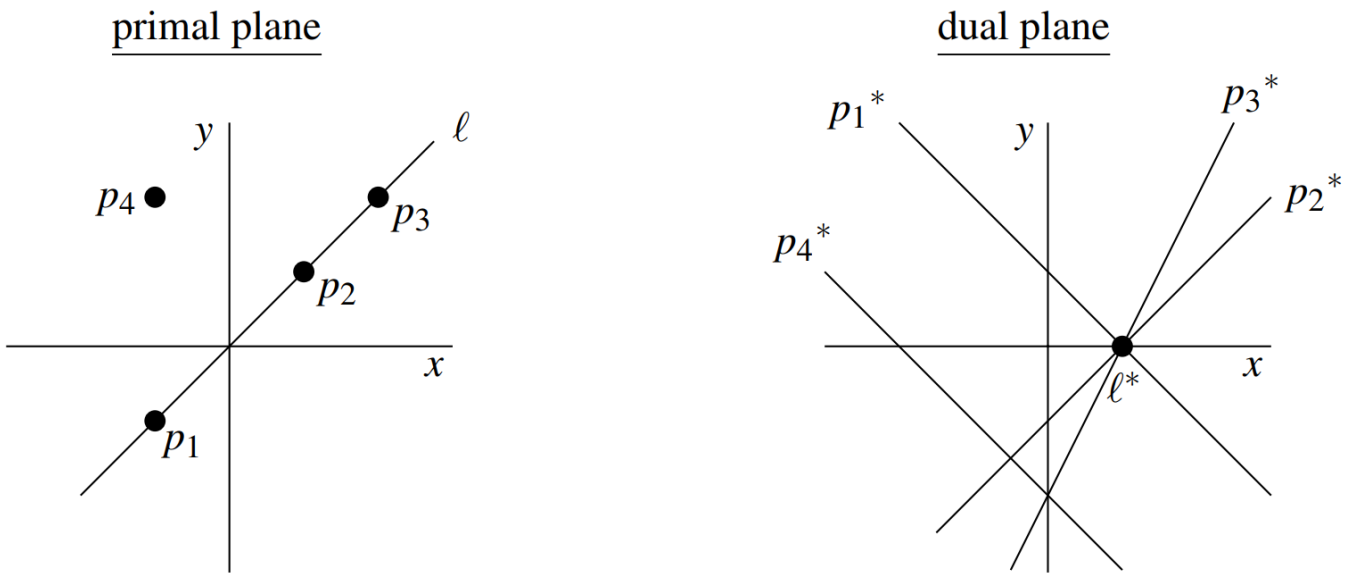
Upiti nad bazom (*range search*): KD/R stabla



Štampane ploče (*windowing*): Intervalna/segmentna stabla

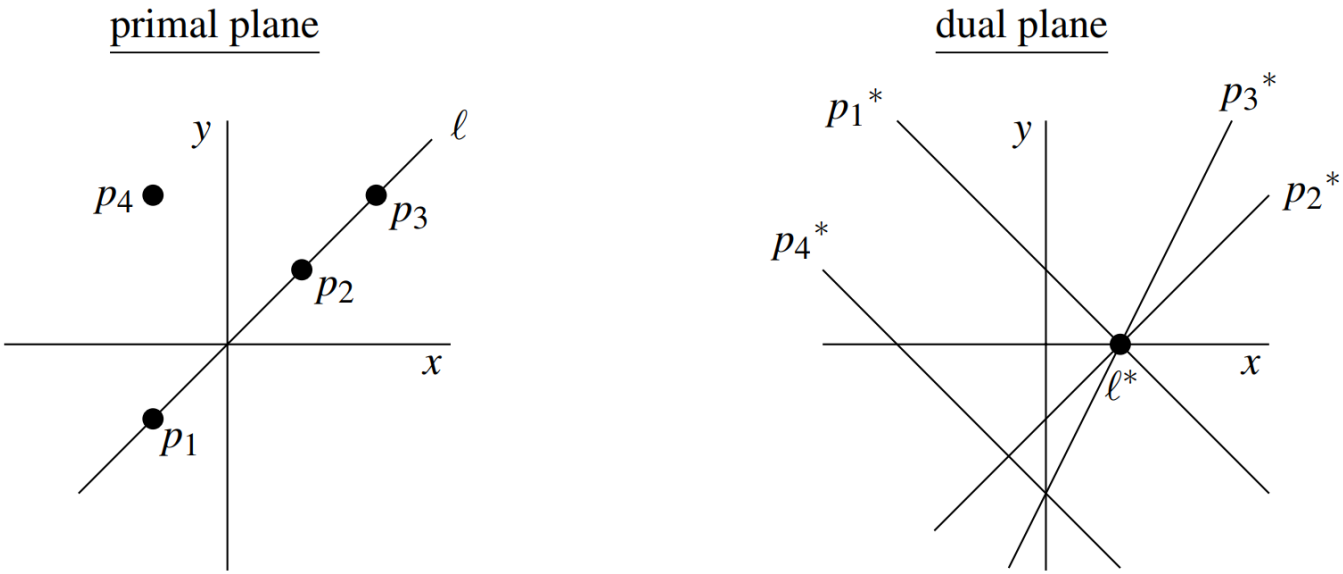


B O N U S: Dualna ravan



- Transformacija ravni kojom dobijamo ekvivalentan ali nekada dosta „poznatiji” problem (poput *inverzije*)
- Slikamo tačku $p(A, B)$ u **pravu** $p^* = Ax - B$
- Slikamo pravu $l = Kx + N$ u **tačku** $l^*(K, -N)$
- Čuva pripadanje, kolinearnost, odnose
- Ne radi za . . .

B O N U S: Dualna ravan



- Transformacija ravni kojom dobijamo ekvivalentan ali nekada dosta „poznatiji” problem (poput *inverzije*)
- Slikamo tačku $p(A, B)$ u **pravu** $p^* = Ax - B$
- Slikamo pravu $l = Kx + N$ u **tačku** $l^*(K, -N)$
- Čuva pripadanje, kolinearnost, odnose
- Ne radi za **vertikalne prave**

Za kraj

Rezime

- Konveksni omotač
- Voronoi dijagram
- *Honourable mentions*

Literatura

- *Computational Geometry - Algorithms and Applications* - Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars

Hvala na pažnji!

- Pitanja?