Problem najmanjšega kroga

Jan Pristovnik in Jan Lampič

5.1.2018

1 Predstavitev problema

Problem najmanjšega kroga ali najmanjšega pokrivnega kroga je matematični probelm izračunavanja najmanjšega kroga, ki vsebuje vse določene množice točk v evklidski ravnini. Če želimo problem rešiti glede na dane točke $P = \{p_1, p_2, .., p_n\}$ v evklidski ravnini, moramo poiskati središče in polmer kroga, ki bo vseboval vse točke iz množice P. Problem najmanjšega kroga je prvotno predlagal angleški matematik Joseph Sylvester leta 1857.
Problem najmanjšega kroga se pojavlja na različnih področjih uporabe, kot so:

- Primer težave z lokacijo objekta, v katerem je treba izbrati lokacijo novega objekta, ki bo zagotavljal storitve vsem v naprej izbranim strankam. Nov objekt mora biti postavljen tako, da bo čim bolj zmanjšal najdaljšo razdaljo, ki jo mora katera koli stranka prepotovati, da doseže novi objekt.

- Za reševanje problemov v okoljski znanosti (načrtovanje in optimizacija topil), prepoznavanje vzorcev (iskanje referenčnih točk), biologija (analiza beljakovin), politična znanost (analiza stranskih spektrov), strojništvo (optimizacija stresa) in računalniška grafika (sledenje žarkov, izločanje) itd.

Najenostavnejši algoritem bi bil, preveriti vse kroge definirane z dvema in tremi točkami, ter preveriti ali vsebuje vse ostale točke in nakoncu izbrati najmanjšega med ustreznimi. Če bi imeli prvotno n točk, bi bilo takih krogov $O(n^3)$. Za vsak krog bi potrebovali $O(n)$, da bi preverili ali vsebuje vse ostale točke, torej skupna časovna zahtevnost bi bila $O(n^4)$. Tak algoritem se je prvič pojavil že okoli leta 1869, od takrat se je zgodilo veliko izboljšav. Danes poznamo veliko praktično uporabnih algoritmov. Teoretično najbolj izpopolnjeni algoritmi imajo časovno zahtevnost $O(n)$.

2 Karakterizacija problema

Večina geometrijskih pristopov za problem išče točke, ki ležijo na meji najmanjšega kroga in temeljijo na naslednjih preprostih dejstvih:

- Najmanjši krog, ki pokriva vse izbrane točke je enoličen.

- Najmanjši krog, ki pokriva vse točke iz množice P lahko določimo z največ tremi točkami, ki ležijo na robu in so vsebovane v P. Če je krog določen samo z dvema točkama, mora razdalja med točkama predstavljati premer minimalnega kroga. Če ga sestavljajo tri točke, potem trikotnik sestavljen iz teh treh točk, ne vsebuje topih kotov.

Kot je pokazal Nimrod Megiddo, je problem iskanja najmanjšega kroga, ki pokriva vse izbrane točke, rešljiv v linearnem času. Enako velja tudi za iskanje najmanjše n-sfere, ki vsebuje vse množice točk v n-dimenzionalnem prostoru.

3 Algoritem Welzla

V najini nalogi sva se osredotočila na algoritem **Emo Welzla**. Algoritem temelji na linearno programskem algoritmu Raimunda Seidela. Izbrala sva ga zaradi njegove enostavnosti in časovne zahtevnosti, ki je pričakovano linearna.

3.1 Ideja algoritma

Kako skonstruiramo najmanjši krog D_n (*angl. disk*), ki vsebuje n-točk ?
Če že poznamo najmanjši krog D_{n-1}, ki pokriva prvih $n-1$ točk $(p_1, \ldots, p_{n-1})$, potem sta za n-to točko možna dva primera.

1. Če točka p_n leži znotraj kroga D_{n-1}, se nič ne spremeni – krog D_{n-1} za točke $p_1, \ldots, p_{n-1}$ je enak kot D_n za točke $p_1, ..., p_n$.
2. Če p_n ne leži znotraj kroga D_{n-1}, je treba izračunati nov krog. Vemo pa, da mora n-ta točka ležati na robu kroga. Torej moramo izračunati najmanjši krog, ki vsebuje točke $p_1, \ldots, p_{n-1}$ in ima p_n na robu.

Ta lastnost skupaj z naslednjimi zahtevki nam omogoča, da najmanjši krog izračunamo na sledeč iterativen način. Naj bo P neprazna množica n točk v ravnini in p točka v P. R pa naj bo množica robnih točk. Potem velja:

- Če obstaja krog, ki vsebuje P in ima R na robu, potem je dobro definiran in enoličen.
- Če p ne leži v krogu $D(P - \{p\}, R)$, potem p leži na robu kroga $D(P, R)$, pod pogojem, da obstaja. To pomeni, da je $D(P, R) = D(P-\{p\}, R \cup \{p\})$.
- Če $D(P, R)$ obstaja, potem obstaja množica S, ki vsebuje $max\{0, 3-|R|\}$ točk v P tako, da je $D(P, R) = D(S, R)$. To pomeni, da je najmanjši krog, ki vsebuje P določen z največ 3 točkami iz P, ki ležijo na robu kroga.

3.2 Algoritem - koda

```
def Welzl(tocke):
    #spremnimo v float in jih nakljucno razvrstimo
    tocke = [(float(x), float(y)) for (x, y) in tocke]
    random.shuffle(tocke)

    #iterativno dodajamo tocke in prepračunavamo krog
    c = None
    for (i,p) in enumerate(tocke):
        if c is None or not je_v_krogu(c,p):
            c = krog1(tocke[: i+1],p)
    return c
```

Definirala sva funkcijo `Welzl`, ki za vhodne podatke sprejme množico točk v ravnini P, vrne pa najmanjši krog, ki vsebuje vse točke iz P. V prvem delu algoritma poskrbiva za pravilen tip podatkov, s funkcijo *random.shuffle* pa poskrbiva za naključno izbiro točk. Za množico točk P izračunamo najmanjši krog postopoma. Začnemo s prazno množico, ki ji zaporedoma dodajamo točke. Za vsako točko preverimo, ali je vsebovana v do sedaj najmanjšem krogu, v tem primeru gremo k naslednji točki. Če točka p ni vsebovana, iz karakterizacije problema vemo, da leži na robu kroga. Nov najmanjši krog izračunamo s pomočjo funkcije `krog1`.

Funkcija `krog1` sprejme do sedaj obravnavane točke in eno robno točko p. Vrne najmanjši krog, ki vsebuje do sedaj obravnavane točke s točko p na robu.

```
#poznamo eno robno točko p
def krog1(tocke,p):
    c = (p[0],p[1],0.0)

    for (i,q) in enumerate(tocke):
        if not je_v_krogu(c,q):#tocke, ki niso v krogu
            if c[2] == 0.0:
                c = premer(p,q)
            else:
                c = krog2(tocke[:i+1],p,q)
    return c
```

Najprej za začetni krog vzamemo kar točko p ter ponovno oblikujemo najmanjši krog. Za vsako točko q preverimo, če ni vsebovana v do sedaj izračunanem najmanjšem krogu. Če je izpolnjen prvi pogoj je nov najmanjši krog enolično določen z dvema točkama p in q - klic funkcije `premer(`p`,`q`)`, ki vrne krog, ki ga določata točki p in q, njuna razdalja pa je kar premer kroga. Sicer vemo, da je točka q na robu in kličemo funkcijo `krog2`. V primeru, da je točka q vsebovana v najmanjšem izračunanem krogu, se krog ne spremeni in nadaljujemo s for zanko.

Funkcija `krog2` sprejme do sedaj obravnavane točke in dve robni točki p ter q. Vrne pa najmanjši krog, ki vsebuje vse obravnavane točke, s točkama p in q na robu. Začetni krog nastavimo s funkcijo `premer(`p, q`)`. Sedaj obravnavamo samo točke, ki niso vsebovane v začetnem krogu. Brez škode za splošnost lahko predpostavimo, da točki p in q ležita na vertikalni premici ℓ. Za vse točke, ki so levo od premice ℓ, poiščemo tisto točko p_l, ki s p in q določa krog s središčem najbolj levo od ℓ. Postopek ponovimo za točke desno od ℓ in dobimo točko p_d. Če so vse obravnavane točke vsebovane v začetnem krogu, se krog ne spremeni. Sicer vrnemo manjšega izmed krogov `ocrtan_krog(`p, q, p_l`)` in `ocrtan_krog(`p, q, p_d`)`, kjer funkcija `ocrtan_krog` vrne očrtan krog določen s

tremi točkami.

4 Eksperiment

4.1 Konstrukcija eksperimenta

Naj bo, za množico točk P, $r(P)$ radij najmanjšega kroga, ki vsebuje P. Naj bo C konveksno območje v ravnini in P_n vzorec n-tih nakjučnih točk iz C. Pri projektu naju je zanimalo:

1. Kako hitro $r(P_n)$ konvergira k $r(P)$?
2. V kolikšni meri je hitrost zgornje konvergence odvisna od oblike območja C?
3. Katera oblika območja C ima najpočasnejšo konvergenco?

Glede na zgornje probleme sva eksperiment zasnovala na sledeč način.

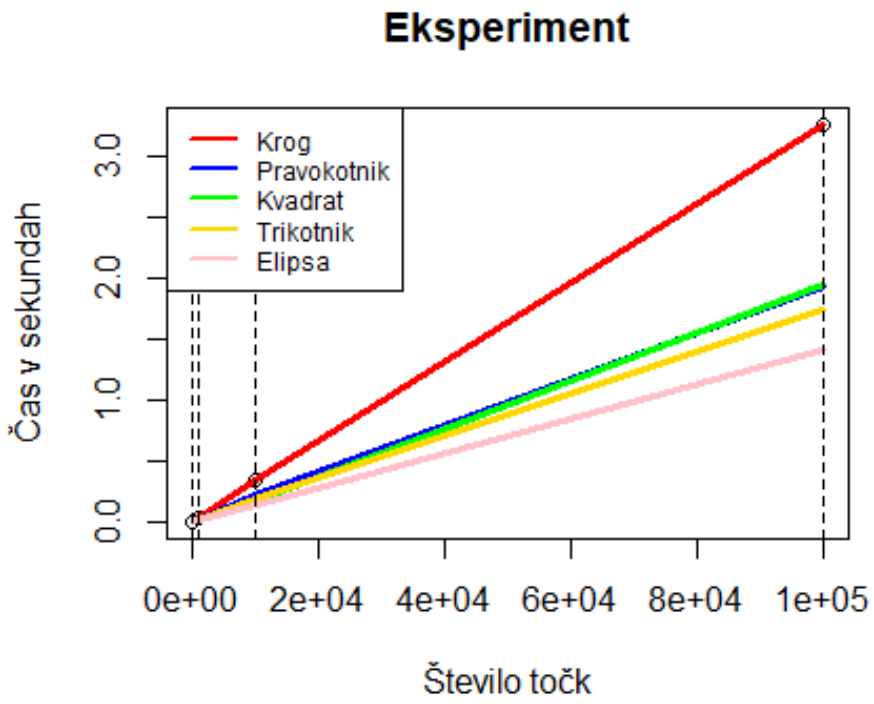
```
def eksperiment(oblike):
    Slovar_casov = {}
    for (i, oblika) in enumerate(oblike):
        casi = []
        for n in (100, 1000,10000,100000):
            cas_n = []
            for _ in range(100):
                tocke = randomTocke(n, oblika)
                to = time.time()
                Welzl.Welzl(tocke)
                t1 = time.time()
                cas_n.append(t1 - to)
            casi.append(sum(cas_n) / 100)

        Slovar_casov[imena[i]] = casi
    return Slovar_casov
```

Izbrala sva si različna konveksna območja krog, elipso, kvadrat, pravokotnik in trikotnik. Za vsako območje sva preverila hitrost konvergence pri različnem številu točk $(10^2, 10^3, 10^4, 10^5)$. Glavni eksperiment sva za vsako območje in izbrano število točk ponovila 100-krat. V vsakem poskusu sva s pomočjo funkcije `randomTocke` zgenerirala točke znotraj območja, na katerih sva izvedla **Welzlov** algoritem ter merila čas konvergence. Nato sva vzela povprečje 100-ih časov in ga shranila v slovar.

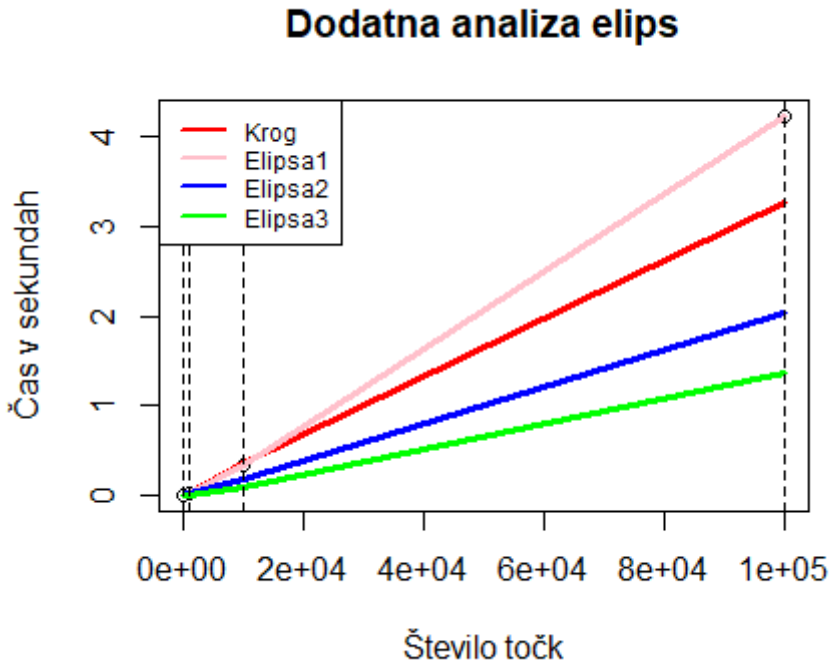
Opomba: Funkcija `randomTocke` za vhodne podatke prejme število točk n ter obliko območja. Funkcija priredi najmanjši pravokotnik, ki vsebuje dano območje. Nato naključno generira točke znotraj pravokotnika in s pomočjo funkcije `je_v_Polygonu` preveri ali je točka vsebovana v iskanem območju. Na koncu vrne n naključnih točk znotraj danega območja. Za elipso in krog sva funkcijo `randomTocke` priredila.

5 Rezultati

Za začetni eksperiment sva primerjala vse oblike med seboj. Isti eksperiment sva ponovila pri različnih velikostih likov. Opazila sva, da sama velikost nima vpliva na rezultate, odstopanja so prisotna zaradi premalega števila poskusov. Največje odstopanje je bilo v vseh poskusih opaziti pri krogu.

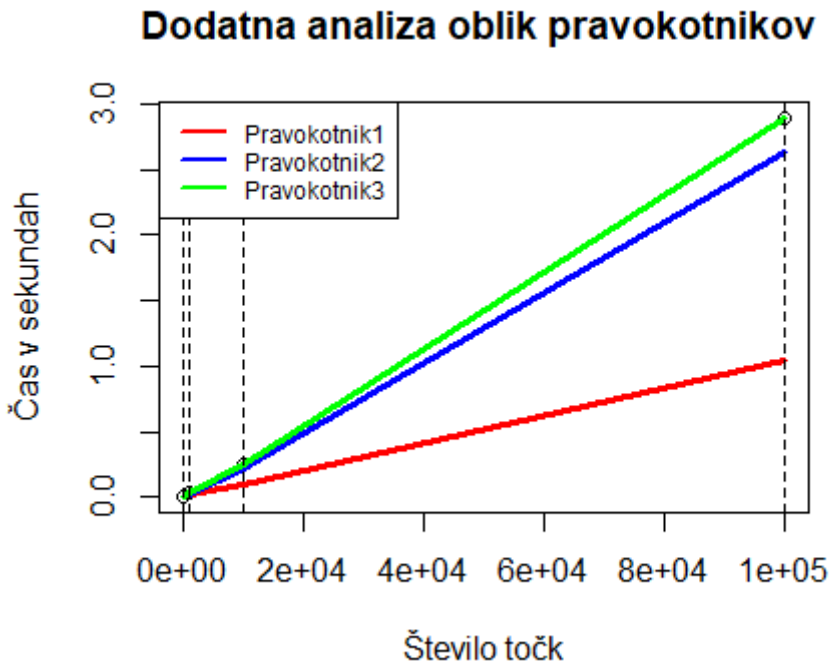


Ker je elipsa imela veliko hitrejšo konvergenco, sva dodatno analizo naredila pri različnih parametrih za elipso v primerjavi s krogom. Naj bo a vodoravna pol os, b navpična pol os elipse. Elipso sva naredila zelo ploščato(Elipsa3), torej $a \gg b$, »običajno«(Elipsa2), $a > b$, in še bolj podobno krogu(Elipsa1), $a \doteq b$. Rezultati so prikazani v spodnjem grafu.



Bolj kot je elipsa ploščata hitrejša je konvergenca, bolj kot je podobna krogu počasnejša je, takšne rezultate sva pričakovala zaradi rezultatov prvega eksperimenta.

Naknadno naju je zanimalo, če bi podobne rezultate dobila tudi z manipuliranjem pravokotnika v primerjavi s kvadratom. Ker se pravokotnik in kvadrat v prvem eksperimentu nista močno razlikovala, v naknadni analizi nisva pričakovala velikih odstopanj. Konvergence so prikazane v spodnjem grafu. Naj bo a vodoravna stranica, b navpična stranica pravokotnika. Pravokotnik1 predstavlja najbolj ploščatega, torej $a \gg b$, Pravokotnik2 »običajnega«,torej $a > b$, in Pravokotnik3 najbolj podobnega kvadratu, $a \doteq b$. Močno odstopanje dobimo le pri najbolj ploščatem pravokotniku, kar se ujema z rezultati različnih elips.



5.1 Povzetek

S pridobljenimi rezultati sva prišla do sledečih ugotovitev:

- Hitrost konvergence Welzlovega algoritma je pričakovano linearna, kar je lepo vidno v rezultatih prikazanih na grafih.
- Oblika območja predstavlja konstantni večkratnik pri linearni konvergenci.
- Najpočasnejšo konvergenco izmed konveksnih območij ima krog, najboljšo pa ploščat pravokotnik.

6 Viri

- Sheng Xu, Robert M. Freund, Jie Sun, Solution Methodologies for the Smallest Enclosing Circle Problem (2003)
- Emo Welzl, Smallest enclosing disks. *New results and New Trends in Computer Science* (1991)
- <https://en.wikipedia.org/wiki/Smallest-circle_problem>[online][5.1.2018]
- <http://code.activestate.com/recipes/66527-finding-the-convex-hull-of-a-set-of-2d-points/>[online][5.1.2018]
- <http://www.cs.uu.nl/docs/vakken/ga/slides4b.pdf>[online][5.1.2018]
- <https://www.nayuki.io/page/smallest-enclosing-circle>[online][5.1.2018]