SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3889

Algoritmi za problem umjetničke galerije

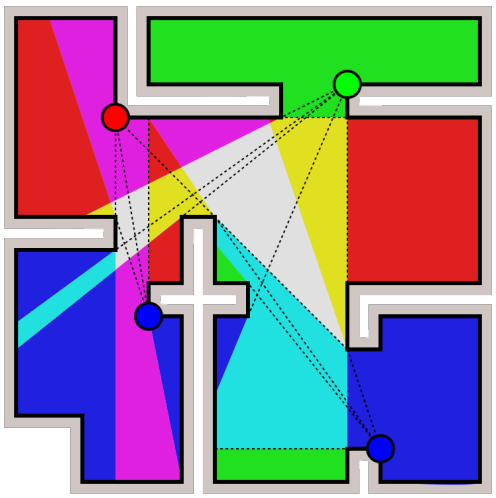
Ivan Bestvina

Zagreb, lipanj 2015.

SADRŽAJ

1. Uvod

Problem umjetničke galerije (engl. *art gallery problem*), ili skraćeno stražarev problem, matematički je problem pronalaska najmanjeg broja čuvara (i njihovih položaja) koji su u mogućnosti nadgledati cjelokupnu površinu umjetničke galerije s n zidova (O'Rourke, 1987). Općenito, problem se može definirati kao traženje najmanjeg skupa točaka S unutar poligona P s n vrhova, takvih da je svaka točka unutar P vidljiva iz barem jedne točke skupa S. Vidljivost između dviju točaka unutar poligona je u ovom kontekstu definirana kao: "dužina koja spaja odabrane točke ne siječe niti jednu stranicu poligona".



Slika 1.1: Primjer rješenja problema umjetničke galerije s 4 čuvara na vrhovima poligona. (Wikipedia, 2014)

Problem je 1973. godine osmislio češki matematičar Victor Klee (O'Rourke, 1987). Nakon prvotnog istraživanja, problem je donekle zaboravljen, sve do kasnih 1980-tih godina, kada Joseph O'Rourke objavljuje knjigu Art Gallery Theorems and Algorithms (O'Rourke, 1987) u kojoj daje pregled starih, ali iznosi i nove algoritme za njegovo rješavanje. Iako prvotno nije postojala jasna vizija gdje bi se u stvarnom svijetu algoritmi ovog problema mogli primijeniti, u zadnje vrijeme sve bržim razvojem robotike

i računalnog vida oni postaju vrlo važni. Jedna od glavnih primjena danas je prilikom određivanja pozicija za tzv. 3D skeniranje prostora – proces koji je vremenski jako zahtjevan pa je svaka optimizacija (smanjenje broja pojedinih skeniranja) korisna (González-Baños, 2001).

Ograničavajući položaje čuvara na rubove ili vrhove poligona, te ograničavajući same poligone na jednostavne (bez rupa), ortogonalne, diskretizirane i sl., dobivamo nekolicinu varijacija problema (O'Rourke, 1987). Ovaj rad fokusira se isključivo na jednostavne poligone s čuvarima na vrhovima (engl. *Point guard art gallery problem*) (Slika 1.1).

Glavni dio rada opisuje moderni algoritam S. K. Ghosha (Ghosh, 2010) uz dvije optimizacije. Algoritam daje rezultate koji su najviše $\mathcal{O}(\log n)$ puta veći od optimalnih (Johnson, 1973) u $\mathcal{O}(n^4)$ vremenu. Dan je opis algoritma te je izvedena njegova a priori vremenska i prostorna složenost (uporabom Steinerovog teorema (Michael, 2009) o broju dijelova na koji n pravaca dijeli plohu). Prokomentirani su implementacijski detalji. Zatim je opisana prva optimizacija koja značajno smanjuje složenost algoritma, nakon čega je dana statistika rezultata testiranja podijeljena po vrstama poligona, te su uspoređeni dobiveni rezultati s teorijski izvedenim predviđanjima. Druga optimizacija koja ubrzava algoritam, ali mu ne smanjuje vremensku složenost, opisana je teorijski.

Kao uvod u ovaj moderni algoritam, nakon kratkog povijesnog i teorijskog pregleda problema, dan je opis dvaju osnovnih algoritama. Prvi od njih, algoritam Avisa i Toussainta (Avis i Toussaint, 1981), direktno proizlazi iz Fiskovog dokaza teorema o minimalnom dovoljnom broju čuvara (Fisk, 1978) koji je opisan u poglavlju 2. Drugi algoritam, minimalni set cover nad triangulacijom, predstavlja kompromis između male složenosti prvog, i suboptimalnih rezultata modernog algoritma S. K. Ghosha. Kao i kod ovog zadnjeg, za oba osnovna algoritma bit će dani detalji oko njihove implementacije, rezultati testiranja algoritama na uzorku od oko 11 000 različitih poligona te usporedba rezultata s predviđanjima.

2. Općenito o problemu umjetničke galerije

2.1. Povijest problema

Godine 1973. na konferenciji na Stanfordu češki matematičar Vašek Chvátal zamolio je svog kolegu Victora Kleea za zanimljiv novi geometrijski problem. Klee je uzvratio problemom koji će kasnije postati poznat kao Problem umjetničke galerije, ili skraćeno Stražarev problem (O'Rourke, 1987). 1975. godine Chvátal objavljuje teorem o minimalnom dovoljnom broju čuvara: za jednostavan poligon od n vrhova uvijek je dovoljno, a nekad i potrebno, $\lfloor n/3 \rfloor$ čuvara (Chvatal, 1975). Detaljan opis i dokaz teorema dani su u sljedećem odlomku.

Nakon nešto više od 10 godina istraživanja, Joseph O'Rourke popularizira problem opširnom knjigom „Art Gallery Theorems and Algorithms" u kojoj, uz presjek gotovo svih tada poznatih algoritama za rješavanje općenitih i specijalnih varijanti problema, pokriva i šira područja matematike i računarstva koja su za ovaj problem vezana (od ovih potonjih poglavlja, danas je u računarstvu vjerojatno najaktualnije poglavlje 7 – Grafovi vidljivosti).

2.2. Minimalni dovoljni broj čuvara

Originalni problem koji je Victor Klee postavio Vašeku Chvátalu 1973. godine nije bio, kao što se u ovom radu razmatra, problem pronalaska efektivnog algoritma za traženje (suboptimalnog) broja i položaja čuvara za zadani poligon, već pronalazak minimalnog dovoljnog broja čuvara za poligon od n vrhova. Drugim riječima, potrebno je pronaći funkciju $g(n)$, takvu da je za svaki poligon od n vrhova dovoljno imati $g(n)$ čuvara, dok je za barem jedan takav poligon $g(n)$ nužan broj čuvara (drugi dio definicije problema nužan je kako bi se izbjegla trivijalna rješenja, npr. $g(n) = n$) (O'Rourke, 1987).

2.2.1. Chvátalov teorem o minimalnom dovoljnom broju čuvara

Par godina nakon što mu je Klee zadao gore opisani problem, Chvátal je objavio svoje rješenje u obliku teorema poznatog pod nazivom Chvátalov teorem umjetničke galerije (Chvatal, 1975):

Teorem 1. *Za pokrivanje jednostavnog poligona od n vrhova uvijek je dovoljno, a nekad i potrebno,* $\lfloor n/3 \rfloor$ *čuvara.*

$$g(n) = \lfloor n/3 \rfloor \tag{2.1}$$

Chvátal je teorem dokazao uz pomoć geometrijskih struktura nazvanih *lepeze*: skup trokuta koji dijele jedan zajednički vrh (O'Rourke, 1987). Očigledno, sve lepeze moguće je pokriti sa samo jednim čuvarom (smještenim na tom zajedničkom vrhu). Dokazivanjem pretpostavke da se svaki poligon može podijeliti na $g \leq \lfloor n/3 \rfloor$ lepeze, dokazan je i originalni teorem.

Jedan od primjera za koji je nužno potrebno $\lfloor n/3 \rfloor$ čuvara je tzv. *češalj* prikazan na slici 2.1.

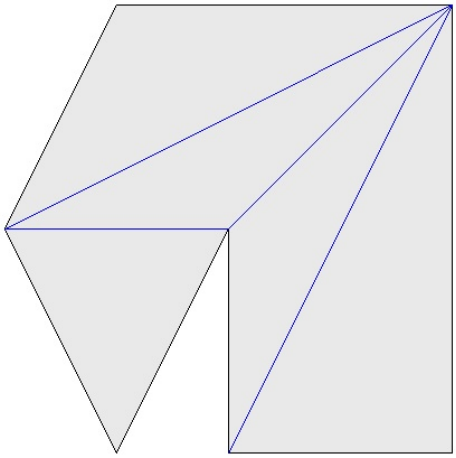
Slika 2.1: Za pokrivanje ovog poligona (*češlja*) od 15 vrhova potrebno je 5 čuvara. (O'Rourke, 1987)

Kao i kasnije opisani Fiskov dokaz, i Chvátal svoj dokaz započinje s triangulacijom poligona. U nastavku je opisan i dokazan teorem o postojanju triangulacije svakog jednostavnog poligona. Detaljan algoritam triangulacije dan je u poglavlju 4.1.

2.2.2. Teorem o postojanju triangulacije jednostavnog poligona

Triangulacija označava podjelu zadanog poligona na trokute koji se međusobno ne preklapaju. Dodatno, u kontekstu problema umjetničke galerije, kada govorimo o triangulaciji, mislimo na triangulaciju bez dodavanja novih vrhova. Drugim riječima, svaka dijagonala koju dodajemo kao stranicu trokuta mora za svoje krajeve imati već postojeće vrhove poligona. Primjer triangulacije prikazan je na slici 2.2.

Teorem 2. *Svaki jednostavni poligon s* n *vrhova moguće je podijeliti na* $n-2$ *trokuta, dodatkom* $n-3$ *unutarnjih dijagonala.*



Slika 2.2: Triangulacija jednostavnog poligona.

Dokaz. Dokaz teorema izvodi se indukcijom po n. Za $n = 3$ teorem je (trivijalno) istinit. Za $n \geq 4$ razmatramo konveksni vrh v_2 (očigledno je kako svaki poligon mora imati barem jedan konveksni vrh) i njemu susjedne vrhove v_1 i v_3 te između njih tražimo unutarnju dijagonalu. Ako je dužina v_1v_3 cijela unutar poligona P (ne siječe niti jednu stranicu poligona P), tada odabiremo dijagonalu $d = v_1v_3$. Inače mora postojati vrh x poligona P koji je unutar trokuta $v_1v_2v_3$. Neka je x onaj vrh unutar tog trokuta koji je, gledano okomito na dužinu v_1v_3, najbliži vrhu v_2. Tada između x i v_2 ne postoji niti jedan drugi vrh poligona P, te je $d = xv_2$.

Poligon P sada je, dijagonalom d, podijeljen na dva dijela, P_1 i P_2. Neka je n_i broj vrhova od P_i. Tada je $n_1 + n_2 = n + 2$, pošto su vrhovi na krajevima dijagonale d sadržani u oba dijela. Kako je $n_i \geq 3$, slijedi da je $n_i < n$, $i = 1, 2$.

Koristeći sada bazu indukcije, proizlazi da, ako će svaki od tih dijelova biti podijeljen na $n_i - 2$ trokuta, ukupni broj trokuta poligona P bit će:

$$(n_1 - 2) + (n_2 - 2) = n - 2 \tag{2.2}$$

Nadalje, ukupni broj dijagonala (dodavanjem dijagonale d) bit će:

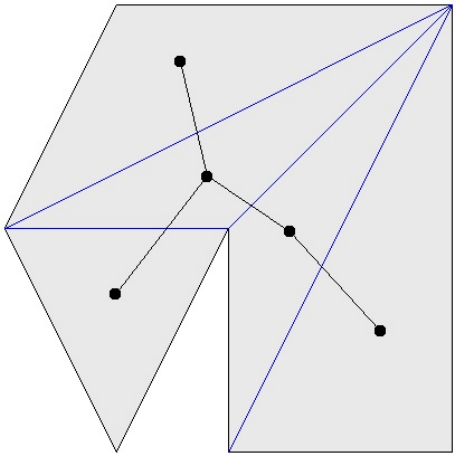
$$(n_1 - 3) + (n_2 - 3) + 1 = n - 3 \tag{2.3}$$

□

Ostatak Chvátalovog dokaza prelazi opseg ovog rada pa je u nastavku dan nešto jednostavniji Fiskov dokaz, iz kojeg, kako je kasnije u radu opisano, proizlazi i prvi algoritam za razmještaj čuvara.

2.2.3. Fiskov dokaz Chvátalova teorema

Neka je zadan poligon P nad kojim je izvršena triangulacija. Neka je zatim konstruiran dualni graf ove triangulacije na sljedeći način: za svaki pojedini trokut konstruiran je jedan vrh grafa, te su povezani međusobno oni vrhovi čiji pripadajući trokuti dijele stranicu (slika 2.3). Očigledno, maksimalan stupanj vrhova takvog grafa je 3 (trokuti imaju 3 stranice).



Slika 2.3: Dualni graf triangulacije – stablo.

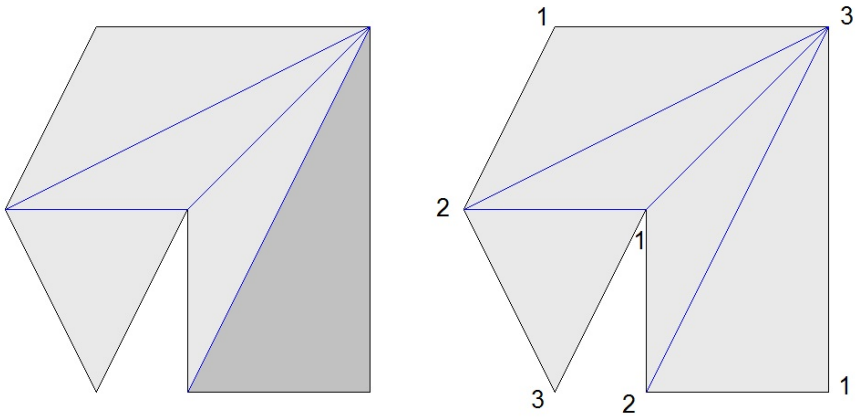
Lema 1. *Dualni graf triangulacije je stablo.*

Dokaz. Pretpostavimo suprotno: dualni graf nije stablo, tj. dualni graf sadrži cikluse. Tada bi taj ciklus, tj. trokuti koji pripadaju vrhovima tog ciklusa, okruživali neke vrhove poligona. Nadalje, to bi značilo da okružuju točke izvan poligona, što bi pak značilo da poligon ima rupe. Takav poligon nije jednostavan poligon pa dolazimo do kontradikcije. Dakle, dobiveni dualni graf triangulacije je stablo. □

Fiskov dokaz bazira se na *tri-bojenju* vrhova triangulacije (slika 2.4). Tri-bojenje je bojenje svakog vrha jednom od triju boja, tako da svaki trokut triangulacije sadrži, na svojim vrhovima, sve tri boje. Ekvivalentno, tri-bojenje možemo definirati i kao bojenje vrhova jednom od triju boja, tako da niti jedan brid triangulacije na svojim krajevima nema obje boje iste.

Algoritam tri-bojenja direktno proizlazi iz dokaza o postojanju tri-bojenja triangulacije jednostavnog poligona.

Lema 2. *Nad trianguliranim jednostavnim poligonom moguće je izvršiti tri-bojanje.*



Slika 2.4: Uho poligona (lijevo) i tri-bojenje poligona označeno brojevima 1, 2 i 3 (desno).

Dokaz. Dokaz o postojanju tri-bojenja triangulacije nad jednostavnim poligonom izvodi se induktivno uz pomoć gore konstruiranog dualnog grafa. Neka je G tako dobiveni graf. Prema gore opisanom dokazu slijedi da je G stablo. Svako stablo, pa tako i G, sadrži barem 2 lista (vrha stupnja 1). Kasnije, kod opisivanja algoritma triangulacije, trokuti koji pripadaju ovim vrhovima grafa nazivat će se *ušima* poligona (slika 2.4). Neka je jedan takav proizvoljno odabrani list uklonjen s grafa. Tako dobiveni graf opet je, očigledno, stablo. Listovi se uklanjaju sve dok ne ostane samo jedan vrh grafa. Vrhovi njemu pripadajućeg trokuta oboje se s tri različite boje. Sada, jedan po jedan, uklonjeni listovi se vraćaju u graf, redoslijedom obrnutim od redoslijeda uklanjanja. Prilikom svakog pojedinog vraćanja nekog vrha grafa, njemu pripadajući trokut spaja se jednom svojom stranicom na neki već obojeni trokut. Tako je boja njegovog trećeg vrha, vrha koji ne pripada toj stranici, jednoznačno određena. Pošto je dualni graf stablo, tj. nema cikluse, nikada se neće dogoditi slučaj „kolizije“ boja, tj. nemogućnost odabira boje u skladu s gore navedenim pravilima. □

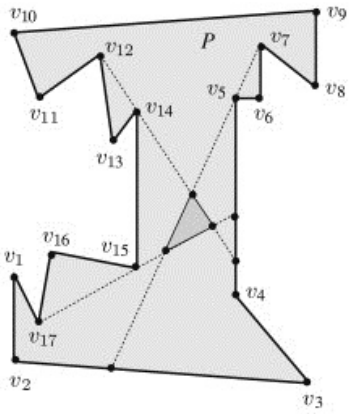
Nakon što je tri-bojenje izvršeno, neka je broj vrhova određene boje označen oznakama a, b i c. Vrijedi $a + b + c = n$. Iz toga slijedi da je $\min(a, b, c) \leq \lfloor n/3 \rfloor$. Pošto svaki trokut triangulacije sadrži po jedan vrh svake od boja, te pošto vrijedi da čuvar u vrhu trokuta pokriva cijeli taj trokut, čuvari raspoređeni po vrhovima jedne od boja nužno će pokrivati sve trokute, tj. cijeli poligon P. Biranjem boje s najmanje vrhova (kako bi se dobio najmanji broj čuvara) slijedi:

$$g(n) = \min\{a, b, c\} \leq \lfloor n/3 \rfloor \tag{2.4}$$

2.3. Podvrste problema

Od objave problema pa do danas nastale su mnoge varijacije inspirirane problemima stvarnog svijeta, ili pak promatranjem originalnog problema u kontekstu istraživačkog područja s kojim prethodno nije bio povezan. Najčešće varijacije problema su varijacije dobivene ograničavanjem pojedinih dijelova definicije. Dva glavna takva ograničenja su ograničenja na vrste promatranih poligona, te ograničenja na položaj i mogućnosti čuvara.

Također, varijacije mogu uključivati i oslabljivanje uvjeta. Na primjer, umjesto uvjeta o pokrivanju cijelog poligona, mogao bi se razmatrati problem pokrivanja samo njegovih rubova. Važno je primijetiti kako ovo stvarno jest prava relaksacija jer postoje konfiguracije poligona i čuvara za koje, iako čuvari nadgledaju sve rubove poligona, ne nadgledaju i cijelu njegovu unutrašnjost. Jedan takav primjer dan je na slici 2.5 – čuvari u vrhovima v_7, v_{12} i v_{17} pokrivaju sve rubove poligona, ali niti jedan od njih ne pokriva trokut označen u sredini.



Slika 2.5: Pokrivanje svih rubova ne osigurava pokrivanje cijelog poligona

2.3.1. Ograničenje vrsta poligona

- Poligoni s rupama
- Jednostavni poligoni (tema ovog rada)
- Ortogonalni poligoni
- Zvjezdasti poligoni
- Spiralni poligoni
- Monotoni poligoni

– Diskretizirani poligoni

2.3.2. Ograničenje položaja i mogućnosti čuvara

– Čuvari unutar poligona (point guards)

– Čuvari na rubovima poligona (edge guards)

– Pomični čuvari na rubovima poligona (točka poligona je vidljiva s odabranog ruba akko je vidljiva s barem jedne točke tog ruba)

– Pomični čuvari unutar poligona

– Čuvari u vrhovima poligona (vertex guards, tema ovog rada)

– Čuvari s ograničenim vidnim poljem: ograničena dubina i/ili kut

2.3.3. Ostale podvrste problema

– Omeđeni 3D prostor umjesto 2D poligona

– Pokrivanje plohe izvan zadanog poligona (tzv. Problem utvrde)

– Pokrivanje plohe i unutar i izvan poligona, s čuvarima na vrhovima poligona (tzv. Problem zatvorskog dvorišta)

2.4. Primjena na probleme iz stvarnog svijeta

Glavna primjena algoritama za rješavanje problema umjetničke galerije je u robotici. Primarno, takvi algoritmi se koriste za pronalaženje optimalne staze nekog robota kroz njegovu okolinu, takve da tijekom prolaženja po stazi robot ima mogućnost „vidjeti“ (skenirati) svaki dio te okoline. Dodatno, algoritmi se u robotici koriste i za statične uređaje, npr. 3D skenere prostora, prilikom određivanja pozicija s kojih je potrebno skenirati zadani prostor. Druge primjene uključuju robotiku u kirurgiji te računalnu grafiku.

3. Općenito o programskoj izvedbi

3.1. Implementacija

3.1.1. Programski jezik i razvojno okruženje

Programsko rješenje izvedeno je u programskom jeziku Java (verzija 8) unutar razvojnog okruženja NetBeans (verzija 8.0.2).

3.1.2. Arhitektura programskog rješenja

Arhitektura programskog rješenja izvedena je u tri sloja: algoritamski sloj, testni sloj te sloj grafičkog sučelja.

Algoritamski sloj sastoji se od 3 razreda implementiranih algoritama: implementacija algoritma Avisa i Toussainta, algoritma minimalni set cover nad triangulacijom te algoritma S. K. Ghosha. Svaki od razreda nasljeđuje apstraktni razred Algoritam, koji definira metode run(), getTime() i getGuards(), te u konstruktoru prima referencu na instancu poligona.

Testni sloj je ujedno i primarni sloj aplikacije, tj. sloj koji komunicira direktno s korisnikom prilikom pokretanja aplikacije preko argumenata komandne linije. Korisnik zadaje datoteku s testnim primjerima te redak unutar te datoteke na kojem se nalazi opis željenog poligona. Testni sloj konstruira instancu poligona iz učitanog znakovnog niza, te slijedno poziva pojedine algoritme, predavajući im referencu na taj poligon. Nakon što su izvršili posao, testni sloj dohvaća testne rezultate (potrebno vrijeme izvršavanja te pronađene čuvare) te ispisuje te rezultate u predefiniranom formatu na standardni izlaz.

Grafički sloj je sloj kojeg referenciraju pojedini algoritmi i testni sloj, kako bi prije, tijekom i nakon izvršavanja prikazivali cjelokupni poligon, pojedine međukorake te krajnji rezultat.

3.2. Testiranje

3.2.1. Automatsko testiranje

Automatsko testiranje algoritama izvršava se uz pomoć skripte koja pokreće aplikaciju po iteracijama, povećavajući svaki puta odabranu liniju unutar datoteke s testnim primjerima za jedan. Također, standardni izlaz preusmjerava se u log datoteku, koja na početku sadrži uzglavlje s popisom algoritama te nazivima pojedinih stupaca podataka koje testni sloj zapisuje.

Ovako izrađena skripta pokreće se na serveru te se log datoteci pristupa iz web browsera kako bi se olakšalo praćenje testiranja, koje je trajalo oko 15 dana.

3.2.2. Testni podaci

Svi testni primjeri korišteni za testiranje implementiranih algoritama preuzeti su sa stranica Brazilskog Instituta za Računarstvo pri Sveučilištu u Campinasu (Couto et al., 2009).

Testni primjeri podijeljeni su u sljedeće dvije kategorije:

- Slučajni ortogonalni poligoni (veličine od 10 do 2500 vrhova) – oko 10 000 primjera
- Slučajni jednostavni poligoni (veličina od 20 do 2500 vrhova) – oko 1000 primjera

Svaki testni primjer predstavlja jedan poligon opisan slijedom njegovih vrhova (u smjeru kazaljke na satu) u obliku „$a/b\ c/d$“ gdje su a/b i c/d razlomci koji definiraju, redom, x i y koordinatu vrha.

3.2.3. Specifikacije sustava na kojem je testiranje izvršeno

Testiranje je izvršeno na servisu DigitalOcean, na sustavu sa sljedećim specifikacijama:

Procesor: Intel Xeon CPU E5-2630L v2, 2.40GHz

RAM: 2GB

OS: Ubuntu 14.04

4. Osnovni algoritmi

U ovom poglavlju dan je pregled dvaju osnovnih algoritama za rješavanje problema umjetničke galerije. Prvi opisani algoritam je algoritam Avisa i Toussainta, koji direktno proizlazi iz Fiskovog dokaza Chvátalovog teorema (Avis i Toussaint, 1981), opisanog u prethodnom poglavlju. Drugi algoritam, minimalni set cover nad triangulacijom, je kombinacija ovog prvog, i modernog algoritma S. K. Ghosha koji je detaljno opisan u sljedećem poglavlju. Ovi osnovni algoritmi, iako daju slabe rezultate, važni su za razumijevanje modernog algoritma koji je osnovna tema ovog rada, te daju korisne referentne podatke za konstruktivnu kvalitativnu ocjenu performansi modernog algoritma.

4.1. Algoritam Avisa i Toussainta

Algoritam Davida Avisa i Godfrieda Toussainta proizlazi direktno iz Fiskovog dokaza Chvátalovog teorema (Fisk, 1978) o problemu umjetničke galerije (Avis i Toussaint, 1981). Algoritam se izvodi u dva odvojena koraka: triangulacija poligona te tri-bojenje dobivene triangulacije. Triangulacija je izvedena tzv. *ear-clipping* metodom (hrv. *rezanje ušiju*) koja počiva na provjeri čine li neka tri (uzastopna) vrha *uho* poligona pa je taj algoritam (nazvan „*isEar*“) zasebno opisan. Uho poligona u ovom kontekstu označava trokut triangulacije koji pripada listu dualnog stabla.

Iako su koraci algoritama bliski koracima Fiskovog dokaza, ovdje su dani iz perspektive samog programskog rješenja, te detaljno opisani u pseudokodu. Teorijska podloga o postojanju triangulacije i tri-bojenja dana je u prethodnom poglavlju pa ovdje neće biti ponavljana.

4.1.1. *IsEar* algoritam

Uho poligona označava 3 uzastopna vrha poligona v_1, v_2 i v_3 za koje vrijedi sljedeće: između vrhova v_1 i v_3 postoji unutrašnja dijagonala poligona. Drugim riječima, du-

žina v_1v_3 cijela je unutar poligona. Nakon triangulacije i konstrukcije dualnog grafa (stabla), ovakve strukture postaju listovi.

Kako bi za dužinu između dva vrha vrijedilo da je unutarnja dijagonala, moraju biti ispunjena dva uvjeta:

- Dužina ne smije biti cijela izvan poligona.
- Dužina ne smije sjeći niti jedan brid poligona.

Prvi se uvjet provjerava računanjem površine s predznakom (engl. *signed area*). Neka su zadani vrhovi a, b i c, te neka $v.x$ i $v.y$ označavaju redom vrijednosti x i y koordinata vrha v. Tada je formula za površinu s predznakom:

$$\left[(a.x \cdot b.y) - (a.y \cdot b.x) + (a.y \cdot c.x) - (a.x \cdot c.y) + (b.x \cdot c.y) - (c.x \cdot b.y)\right]/2 \tag{4.1}$$

Pošto dijeljenje s 2 ne utječe na predznak, može se zanemariti. Ako je tako dobivena površina pozitivna, dužina nije cijela izvan poligona. Ako je negativna, dužina nije cijela unutar poligona. Dakle, ako je površina negativna, dužina je ili cijela, ili dijelom izvan poligona, te algoritam vraća vrijednost *false* (dužina nije unutarnja dijagonala).

Inače, algoritam provjerava drugi uvjet prolazeći redom kroz sve bridove poligona. Pronađe li brid koji siječe zadanu dužinu, prekida traženje i vraća *false*. Ne pronađe li niti jedan takav brid, vraća *true*.

4.1.2. Triangulacija *ear-clipping* metodom

Triangulacija se izvodi unutar jedne *while* petlje, koja terminira nakon što je broj pronađenih dijagonala triangulacije dostigao $n - 3$. Pošto nam lista dijagonala triangulacije za algoritam Avisa i Toussainta nije bitna, pronađene trokute odmah dodajemo u dualni graf.

U opisu algoritma, $P[i]$ označava vrh poligona s indeksom i (za prvi vrh vrijed $i = 0$), dok se oznake $P[i + 1]$ te $P[i - 1]$ odnose redom na sljedeći i prethodni vrh, gledano kružno. Drugim riječima, pretpostavlja se implicitna operacija $\bmod n$ nad svim traženim indeksima.

Algoritam 1 Triangulacija *ear-clipping* metodom

1: **procedure** EARCLIPTRIANGULATE(P) ▷ P je zadani poligon
2: $d \leftarrow 0$ ▷ Broj pronađenih dijagonala
3: $i \leftarrow 0$ ▷ Indeks trenutnog vrha
4: $n \leftarrow |P|$ ▷ Veličina poligona
5: $DG \leftarrow \emptyset$ ▷ Dualni graf triangulacije

```
    while d < (n − 3) do                ▷ Konačni broj dijagonala je n − 3 prema (2.3)
        if isEar(P[i − 1], P[i], P[i + 1]) then        ▷ Ako je trokut uho poligona
            DG ← DG ∪ {(P[i − 1], P[i], P[i + 1])}          ▷ Dodaj ga u DG
            P ← P \ {P[i]}                                  ▷ Ukloni vrhi P[i]
            d ← d + 1
        end if
        i ← i + 1
    end while
    DG ← DG ∪ {P}                ▷ Preostala su 3 vrha u P koji čine zadnji trokut
    return DG
end procedure
```

Tijekom trianguliranja dobiveni trokuti direktno su spremani u dualni graf DG koji je stablo (Lema 1).

4.1.3. Tri-bojenje

Rezultat prethodnog koraka algoritma je stablo DG – dualni graf triangulacije. Tri-bojenje triangulacije, uz pomoć dobivenog dualnog grafa, izvodi se algoritmom ekvivalentnim onom opisanom kod Fiskovog dokaza (Fisk, 1978) Chvátalovog teorema:

Algoritam 2 Tri-bojanje triangulacije

```
procedure THREECOLOUR(DG)
    S ← ∅                                  ▷ Inicijalizacija praznog stoga S
    while DG ≠ ∅ do
        Neka je l list u DG
        push l onto S
        G ← G \ {l}                                  ▷ Premjesti l iz DG u S
    end while
    pop l from S
    Oboji vrhove trokuta l različitim bojama
    while S ≠ ∅ do
        pop l from S
        Oboji jedini neobojani vrh trokuta l preostalom od tri boje
    end while
end procedure
```

Već prilikom tri-bojenja korisno je brojati koliko je puta korištena određena boja. Nakon bojenja odabire se najmanje puta korištena boja te se na vrhove te boje dodaju čuvari.

4.1.4. Vremenska složenost

Vremenska složenost algoritma računa se odvojeno po dijelovima algoritma. Složenost *isEar* metode je $\mathcal{O}(n)$, u slučaju da promatrana dužina nije cijela izvan poligona, pošto algoritam za svaki od n bridova ispituje sijeku li se s tom dužinom.

Nadalje, sama triangulacija poziva *isEar* metodu sve dok ne pronađe sve trokute, prolazeći poligonom u krug. U najgorem slučaju sljedeće uho nalazit će se uvijek neposredno prije trenutne pozicije, te će algoritam morati proći sve dosad neodrezane vrhove. U tom slučaju, $isEar$ se poziva $n + (n-1) + (n-2) + \ldots + 5 + 4 = 1/2(n-3)(n+4)$, pošto se nakon svakog pronalaska uha broj vrhova smanji za 1, sve dok $n \neq 3$ (kada preostaje samo jedan, zadnji trokut, dodan u 14. koraku triangulacije). U prosječnom slučaju, uzimajući u obzir činjenicu da svi poligoni nužno imaju barem dva uha, dok ih većina ima i puno više, ova složenost smanjuje se za neki konstantni faktor (koji, primarno, ovisi o vrsti poligona). Općenito, složenost u $\mathcal{O}$-notaciji iznosi $\mathcal{O}(n^2)$. Složenost tri-bojenja je linearna: $\mathcal{O}(n)$. Zajedno, složenost cijele triangulacije je

$$\mathcal{O}(n + n^2 + n) = \mathcal{O}(n^2)^1 \tag{4.2}$$

4.1.5. Prostorna složenost

U algoritmu Avisa i Toussainta dvije su prostorno značajne podatkovne strukture: graf i stog. Pošto se oba ispunjavaju trokutima triangulacije (kod grafa predstavljenima kao vrhovi grafa, kod stoga kao skupovi triju točaka), te se ti trokuti iz grafa premještaju, jedan po jedan, na stog, njihova ukupna prostorna složenost je konstantna te iznosi $\mathcal{O}(n)$. Kako u algoritmu nema drugih prostorno značajnih struktura, prostorna složenost cijelog algoritma je $\mathcal{O}(n)$.

4.1.6. Implementacijski detalji *isEar* metode

Implementacijski zanimljiv dio algoritma zasigurno je $isEar(a, b, c)$ metoda koja prima tri vrha poligona te ispituje čine li te tri točke uho poligona. Kao što je gore opi-

[1] Postoje algoritmi koji triangulaciju izvode u $\mathcal{O}(n \log n)$ (Garey et al., 1978), te u $\mathcal{O}(n \log \log n)$ (Tarjan i Van Wyk, 1988).

sano, ona se bazira na metodi $clockwise(a, b, c)$ koja ispituje jesu li neke tri točke u ravnini orijentirane u smjeru kazaljke na satu. Ta metoda se pak bazira na metodi $signedArea(a, b, c)$ koja računa površinu trokuta s predznakom.

Metoda *isEar* u teoriji jednoznačno određuje je li dijagonala dijelom izvan, ili dijelom unutar poligona, sve dok tri zadane točke nisu kolinearne (kada je površina $A = 0$). Ipak, kod diskretiziranih vrijednosti na računalu, ekstremno male površine također se u memoriji preslikavaju u vrijednost 0. Zbog toga je nužno implementirati dodatni ispravljački dio algoritma *clockwise* kako bi metoda funkcionirala ispravno. Ispravno funkcioniranje u ovom slučaju znači: za svake tri nekolinearne točke a, b i c vrijedi $clockwise(a, b, c) = \neg clockwise(c, b, a)$.

Ispravljački dio algoritma uvodi malu pozitivnu konstantu ε (u konkretnom slučaju ove implementacije $\varepsilon = 10^{-10}$) te se oslanja na proizvoljno implementiranu relaciju linearnog uređaja nad točkama $pointCompare$ (skraćeno pC). Drugim riječima, ovakva relacija mora zadovoljavati sljedeća tri uvjeta:

Linearnost: $pC(a, b) \vee pC(b, a)$

Antisimetričnost: $pC(a, b) \wedge pC(b, a) \rightarrow a = b$

Tranzitivnost: $pC(a, b) \wedge pC(b, c) \rightarrow pC(a, c)$

Dodatno, ako su zadane različite točke a i b, iz antisimetričnosti slijedi $\neg(pC(a, b) \wedge pC(b, a))$. U kombinaciji s linearnošću, slijedi:

$$pC(a, b) \iff \neg pC(b, a) \tag{4.3}$$

Programsko rješenje sadrži istoimenu metodu $pC(a, b)$ koja implementira karakterističnu funkciju relacije koja zadovoljava gore opisane uvjete. Zbog strukture algoritma točke a i b koje zadajemo metodi uvijek su različite. Zbog toga za metodu $pC(a, b)$ vrijedi:

$$pC(a, b) = \neg pC(b, a) \tag{4.4}$$

U ovoj konkretnoj implementaciji metoda prvo uspoređuje točke po x koordinati pa, ako su različite, vraća rezultat te usporedbe, inače vraća rezultat usporedbe po y koordinati.

Nakon ovako definirane konstante ε i relacije $pointCompare$, ispravljeni algoritam $clockwise(a, b, c)$ radi na sljedeći način:

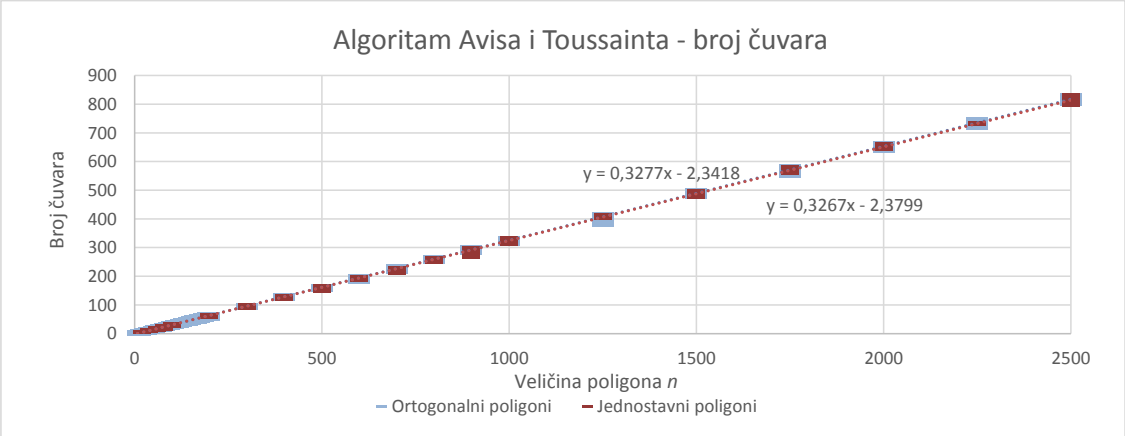
Algoritam 3 Određivanje orijentacije niza od tri točke

1: **procedure** CLOCKWISE(a, b, c)
2: $A \leftarrow signedArea(a, b, c)$
3: **if** $-\varepsilon < A < \varepsilon$ **then**
4: $br \leftarrow 0$
5: **if** $pointCompare(a, b)$ **then** $br \leftarrow br + 1$ **end if**
6: **if** $pointCompare(b, c)$ **then** $br \leftarrow br + 1$ **end if**
7: **if** $pointCompare(c, a)$ **then** $br \leftarrow br + 1$ **end if**
8: **return** $(br = 2)$
9: **end if**
10: **return** $A > 0$
11: **end procedure**

Koraci 2 – 9 su ispravljački dijelovi algoritma. Ako je dobivena površina između $-\varepsilon$ i ε, njezin predznak se zanemaruje (jer zbog diskretizacije nije značajan). U tom slučaju svejedno je, što se geometrijskih svojstava tiče, hoće li algoritam zaključiti da su točke poredane u smjeru kazaljke na satu ili suprotno. Ipak, nužno je osigurati da metoda *clockwise*, bude li pozvana s istim parametrima zadanima suprotnim redoslijedom, vrati komplement prvotne vrijednosti. Redoslijed ovdje označava orijentaciju niza točaka promatranih kružno. Drugim riječima, promatramo li točke a, b i c, u jednu orijentaciju spadaju redoslijedi abc, bca, cab, a u drugu cba, bac, acb. Zbog gore opisanih svojstava, od tri poziva metode *pointCompare*, bit će ispunjen jedan ili dva. Očigledno, sve permutacije iste orijentacije zadovoljit će isti broj poziva, dok će permutacije suprotne orijentacije imati komplementarne rezultate, što je i željeno ponašanje.

4.1.7. Rezultati testiranja

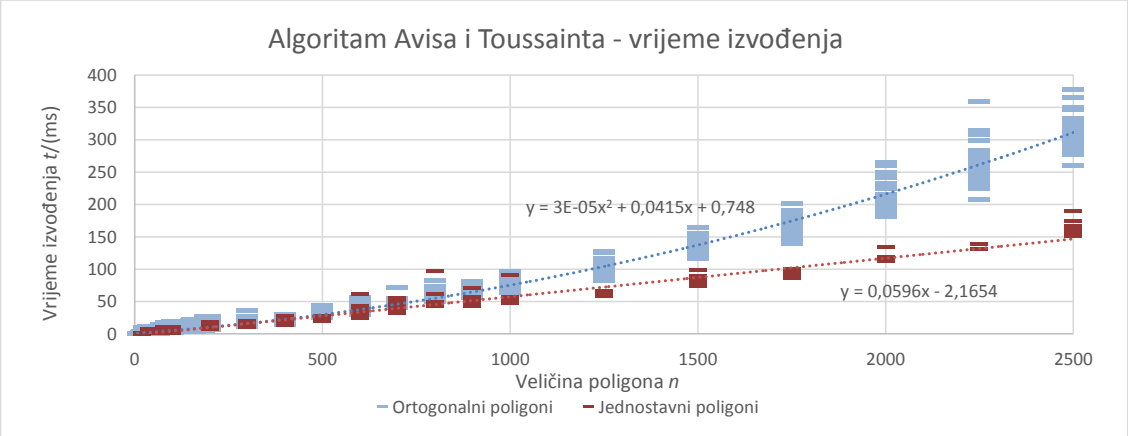
Broj čuvara



Slika 4.1: Broj čuvara algoritma Avisa i Toussainta.

Rezultati algoritma približno su jednaki za obje vrste poligona. Kao što je i predviđeno, algoritam pronalazi $\approx \lfloor n/3 \rfloor$ čuvara (slika 4.1).

Vrijeme izvođenja



Slika 4.2: Vrijeme izvođenja algoritma Avisa i Toussainta.

Vrijeme izvođenja algoritma Avisa i Toussainta, za ortogonalne poligone, približno je predviđanjima, iako je kvadratni koeficijent iznimno mali. S druge strane, jednostavni poligoni izvode se značajno brže te je najbolja aproksimacija tog skupa podataka linearna funkcija (slika 4.2).

Iako je ovakav rezultat u suprotnosti s predviđanjima, on je lako objašnjiv. Naime, kod slučajno generiranih jednostavnih poligona, *isEar* funkcija u većini slučajeva prima vrhove čija dijagonala nije cijela unutar poligona. U tim slučajevima njezina složenost je $\mathcal{O}(1)$, što objašnjava dobivene rezultate.

4.2. Minimalni set cover nad triangulacijom

Algoritam *minimalni set cover nad triangulacijom* predstavlja neku vrstu međukoraka između druga dva opisana algoritma. Iako se u praksi ne koristi, uključen je u ovaj rad kao dobar izvor testnih rezultata koji u kombinaciji s rezultatima algoritma Avisa i Toussainta daju koristan kontekst za tumačenje rezultata modernog algoritma.

4.2.1. Opis algoritma

Kao i prethodni algoritam, i ovaj započinje s triangulacijom poligona. Kako je algoritam triangulacije isti[2] kao i u prethodnom poglavlju, njegov opis neće biti ponavljan.

Nakon triangulacije, čiji je rezultat ovaj put skup trokuta, a ne dualni graf, slijedi pronalazak položaja čuvara uporabom tzv. greedy set cover algoritma.

4.2.2. *Greedy* set cover algoritam

Minimalni set cover problem jedan je od najpoznatijih *NP-potpunih* problema. Za zadani skup elemenata $U = \{1, 2, \ldots, m\}$ (engl. *universe*) i skup od n podskupova S čija unija daje U, potrebno je pronaći najmanji podskup od S za koji je unija također jednaka U.

Greedy (hrv. *pohlepni*) algoritam set cover problema iznimno je jednostavan, ali za većinu zadanih problema i iznenađujuće efektivan, zbog čega ne čudi njegova popularnost. Algoritam započinje s praznim rezultantnim skupom G kojeg iterativno popunjava s elementima skupa S, odabirući ih po pohlepnoj heuristici: "Uzmi onaj skup iz S koji donosi najviše dosad nepokrivenih elemenata." Algoritam terminira kada unija skupova u G postane jednaka U.

U kontekstu problema umjetničke galerije i triangulacije poligona, algoritam se formulira na sljedeći način:

[2] Jedina razlika je što se ovaj put ne konstruira dualni graf, već lista trokuta.

Algoritam 4 Oblikovanje minimalnog set cover problema nad triangulacijom

1: **procedure** TRIANGULATIONSETCOVER(P, U) ▷ U – skup trokuta triangulacije
2: $S \leftarrow \emptyset$
3: **for all** $v \in P$ **do**
4: $T \leftarrow \emptyset$
5: **for all** $t \in U$ **do**
6: **if** $v \in t$ **then**
7: $T \leftarrow T \cup \{t\}$
8: **end if**
9: **end for**
10: $S \leftarrow S \cup \{T\}$
11: **end for**
(početak *greedy* algoritma)
12: $G \leftarrow \emptyset$
13: **while** $\bigcup G \neq U$ **do**
14: $s \leftarrow argmax_{T \in S}\{|T|\}$ ▷ Najveći skup u S
15: $G \leftarrow G \cup \{s\}$
16: Dodaj čuvara na vrh koji odgovara skupu s.
17: $S \leftarrow S \setminus \{s\}$ ▷ Optimizacijski korak – nije nužan.
18: **end while**
19: **end procedure**

Korak 15 je u potpunosti optimizacijski, te se kod teorijskog razmatranja može zanemariti.

Za poligon od n vrhova, skup S biti će veličine n. Kako je svaki trokut obrađen tri puta (po jednom za svaki od njegovih vrhova) te kako ima $n - 2$ trokuta, zbroj svih veličina skupova u S iznosi $3(n - 2) \approx 3n$. Iz ovoga proizlazi kako prosječna veličina skupa u S iznosi $3n/n = 3$.

Iz ovoga se pak može pretpostaviti kako će gornja granica broja čuvara skoro uvijek biti $n/3$. Ipak, kao što će iz rezultata testiranja biti vidljivo, algoritam daje bolje rezultate od prethodno opisanog te se ova granica u pravilu vrlo rijetko dostiže. Važno je napomenuti kako bi teoretski broj čuvara ovog algoritma ipak mogao biti veći od $n/3$.

4.2.3. Vremenska složenost

Vremenska složenost cijelog algoritma ovisi o vremenskoj složenosti triangulacije, koja je već izvedena i iznosi $\mathcal{O}(n^2)$, i vremenskoj složenosti greedy algoritma.

Korak 7 ponavlja se za svaki vrh svakog trokuta, dakle $\approx 3n$ puta. Kako bi našao traženi skup u S u koraku 14, algoritam mora proći kroz sve skupove skupa S, te prebrojati sve još neobrađene elemente tih skupova. Prosječna veličina skupa u S je 3 (kako je pokazano u prethodnom ulomku). Pretpostavimo li da je (u praktično najgorem slučaju) broj tih iteracija (broj čuvara) $n/3$, dobivamo vremensku složenost od:

$$\mathcal{O}((n/3) \cdot 3n) = \mathcal{O}(n^2) \tag{4.5}$$

Iskoristimo li optimizaciju u koraku 17, ova vremenska složenost smanjit će se samo za mali konstantni faktor. U svakom slučaju, složenost triangulacije od $\mathcal{O}(n^2)$ zadržat će ukupnu složenost nepromijenjenom.

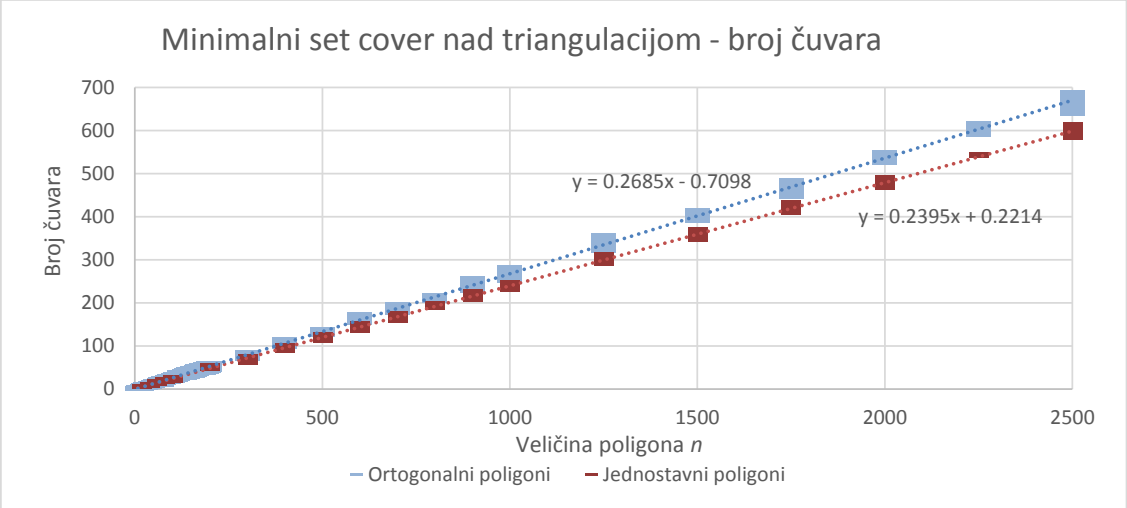
4.2.4. Prostorna složenost

Prostorna složenost triangulacije je $\mathcal{O}(n)$. Složenost skupa U je također $\mathcal{O}(n)$, složenost skupa S je $\mathcal{O}(3n)$, a složenost skupa G je $\mathcal{O}(n)$. Sve ove vrijednosti izvedene su u prethodnim poglavljima. Slijedi da je prostorna složenost cijelog greedy set cover algoritma $\mathcal{O}(n)$, pa je prostorna složenost algoritma minimalni set cover nad triangulacijom također $\mathcal{O}(n)$.

4.2.5. Rezultati testiranja

Broj čuvara

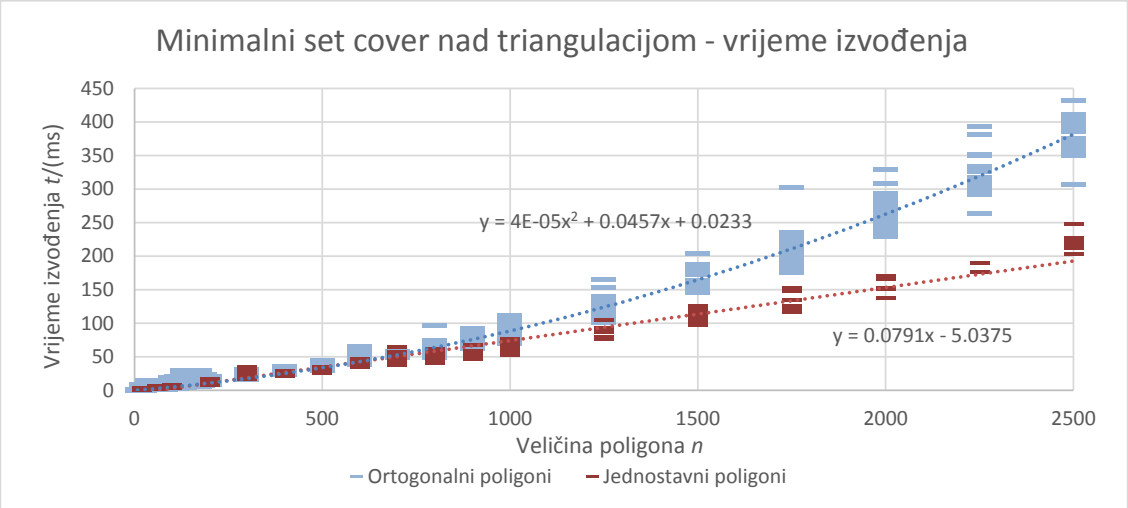
Rezultati algoritma u skladu su s predviđenim vrijednostima. U usporedbi s algoritmom Avisa i Toussainta, za obje vrste poligona vidljivo je poboljšanje. Ipak, kod ovog algoritma postoji razlika između ortogonalnih i jednostavnih poligona. Ova razlika može se objasniti većom uniformnošću ortogonalnih poligona, što uzrokuje ravnomjerniju raspodjelu trokuta triangulacije po vrhovima poligona, tj. po lepezama tih vrhova.



Slika 4.3: Broj čuvara algoritma minimalni set cover nad triangulacijom.

Vrijeme izvođenja

Kako je triangulacija vremenski najzahtjevniji dio ovog algoritma, kao i algoritma Avisa i Toussainta, očekivano je da će im vremena izvođenja biti slična, što testni podaci i prikazuju. Kao i kod prethodnog algoritma, kvadratni član polinoma i ovdje je iznimno malen za ortogonalne, a beznačajan za jednostavne poligone.



Slika 4.4: Vrijeme izvođenja algoritma minimalni set cover nad triangulacijom.

5. Algoritam S. K. Ghosha

Aproksimacijski algoritam Subira Kumara Ghosha, objavljen 2008. godine[1] (Ghosh, 2010), jedan je od najnovijih aproksimacijskih algoritama za rješavanje problema umjetničke galerije. Algoritam se, kao i prethodno opisani, bazira na pretvaranju problema umjetničke galerije u minimalni set cover problem particijom poligona. Ipak, za razliku od prethodnog algoritma, ovaj za particiju ne koristi triangulaciju, već podjelu poligona na posebno oblikovane konveksne regije nazvane konveksnim komponentama. Nakon ovakve particije, algoritam razrješava problem upotrebom greedy set cover algoritma, detaljno opisanog u kontekstu prethodnog algoritma.

5.1. Opis algoritma

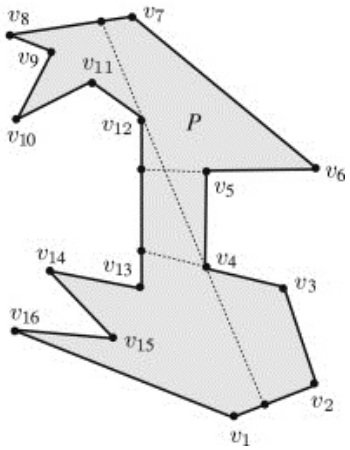
Motivacija algoritma bila je pronaći efektivnu pretvorbu problema umjetničke galerije u minimalni set cover problem. Drugim riječima, pronaći algoritam za konstrukciju particije poligona koja, promatrana kao metoda diskretizacije tog poligona, s jedne strane osigurava dobre performanse set cover algoritama kojima se prosljeđuje, a s druge strane, u konačnici, omogućava tom set cover algoritmu suboptimalne rezultate.

Kao prirodna particija za ovu primjenu u početku se činila ona dobivena rezanjem poligona produžecima njegovih bridova (Slika 5.1) (Ghosh, 2010).

Ipak, kako bi pretvorba u minimalni set cover problem bila što smislenija, particija poligona trebala bi zadovoljavati sljedeći uvjet: dijelovi particije su ili cijeli vidljivi, ili cijeli nevidljivi iz svih pojedinih vrhova poligona. Očigledno je kako gore opisana particija ovaj uvjet ne zadovoljava. Također, iz slike 5.1 je vidljivo kako particija produžecima rubova za pokrivanje zahtjeva tri lepeze iz vrhova v_1, v_4 i v_7, dok particija opisana u nastavku (produžecima rubova i dijagonala) zahtjeva samo dvije lepeze: v_1 i v_7.

Ghosh predlaže particiju dobivenu rezanjem poligona ne samo produžecima svih bridova, već i svim unutarnjim dijagonalama i njihovim produžecima. Produžeci se

[1] doi:10.1016/j.dam.2009.12.004



Slika 5.1: Usporedba korištene particije i particije produženim rubovima. Za pokrivanje poligona dovoljni su čuvari u vrhovima v_1 i v_7. (Ghosh, 2010)

protežu sve do prvog sjecišta s nekim bridom poligona. Dobivene konveksne regije poligona Ghosh naziva *konveksnim komponentama*.

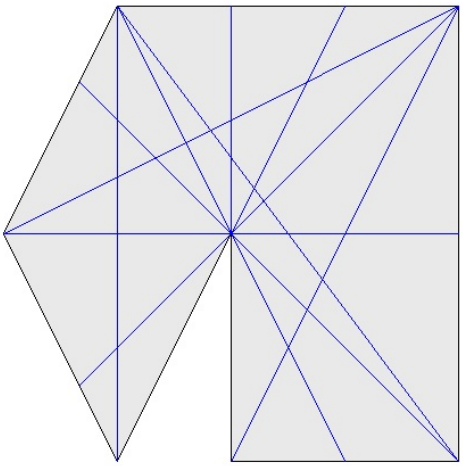
Formalnije, konveksna komponenta particije P je svaki $c \subset P$ kroz koji ne prolazi niti jedna dužina $d \subset P$, a koja prolazi kroz barem dva vrha od P (Ghosh, 2010).

U nastavku rada sve ove dužine kojima siječemo poligon nazivat ćemo *tetivama*.

Kako se tetive protežu između rubova poligona, moguće ih je promatrati kao pravce na tom dijelu ravnine (kasnije će ovo svojstvo biti korišteno kod izvođenja vremenske složenosti konstrukcije tih dužina). Iz ovoga proizlazi da će kutovi na sjecištima tetiva nužno biti konveksni. Kako se svaki konkavni vrh poligona dijeli, produžecima rubova, na tri konveksna kuta, proizlazi da su svi kutovi ovakve particije konveksni, tj. da gore opisane konveksne komponente uistinu jesu konveksni poligoni. Primjer ovakve particije prikazan je na slici 5.2

Korisno je u kontekstu ovakve particije redefinirati pojam *lepeze* koji je uveden kod Chvátalova teorema. Lepeza vrha v particije je skup onih konveksnih komponenti koje su (cijele) vidljive iz v. Zbog gore opisanog pravila, nije potrebno eksplicirati nužnost potpune vidljivosti konveksne komponente – ako je vidljiva neka točka komponente, vidljiva je i cijela komponenta.

Nakon ovako izvršene particije za svaki se vrh konstruiraju skupovi konveksnih komponenata vidljivi iz tog vrha, koji se zatim promatraju kao minimalni set cover problem, na način ekvivalentan minimalnom set cover problemu nad triangulacijom.



Slika 5.2: Particija poligona dijagonalama i produžecima rubova i dijagonala.

5.2. Pseudokod algoritma

5.2.1. Konstrukcija rubova konveksnih komponenti

Konstrukcija tetiva, dužina na kojima će ležati rubovi svih konveksnih komponenti, sastoji se od dva dijela: konstrukcija produžetaka rubova te konstrukcija produženih dijagonala. Produžene dijagonale zapravo su sastavljene od više zasebnih dužina: osnovne dijagonale te njezinih produžetaka u jednu ili obje strane.

Konstrukcija produžetaka rubova

Algoritam 5 Konstrukcija produžetaka rubova

1: **procedure** EDGESEXTEND(P)
2: **for all** $v_a v_b \in P$ **do** ▷ $v_a v_b$ označava neki rub poligona P
3: Neka je p pravac određen točkama v_a i v_b.
4: **for** v_i, $i = a, b$, ako je v_i konkavan vrh **do**
5: Neka je rub e najbliži vrhu v_i, takav da e siječe p.
6: Dodaj dužinu između v_i i sjecišta e i p u listu produžetaka rubova.
7: **end for**
8: **end for**
9: **end procedure**

Važno je primijetiti kako koraci 5 i 6 zapravo uključuju tri različite situacije, koje je kod implementacije ovog algoritma nužno dobro razumjeti:

– Pravac p uistinu siječe rub e.

– Rub e leži na pravcu p. Za sjecište se uzima onaj kraj od e koji je bliži vrhu v_i.

– Rub e jednim svojim krajem dodiruje pravac p. Neka je taj kraj v_e.

- Ako se rubovi s obje strane v_e (rub e i njemu susjedni) nalaze na istoj strani pravca p, sjecište v_e se zanemaruje i pretraga se nastavlja.
- Inače, za sjecište se uzima v_e.

Konstrukcija produženih dijagonala

Konstrukcija produženih dijagonala izvodi se u dva koraka. Prvo se konstruiraju sve unutarnje dijagonale, a zatim se konstruiraju produžeci. Kod konstrukcije dijagonala, za svaki par vrhova provjerava se, na način ekvivalentan onom kod triangulacije u poglavlju 4.1.1, postoji li unutarnja dijagonala između njih.

Nakon konstrukcije svih unutarnjih dijagonala, konstrukcija produžetaka izvodi se slično produžecima rubova – dijagonala i njoj priležeći rubovi mogu se promatrati kao susjedni rubovi kod gore opisanog algoritma 5. Specijalni slučajevi opisani kod konstrukcije produžetaka rubova ekvivalentno se pojavljuju i razrješavaju i ovdje.

5.2.2. Particija poligona po konveksnim komponentama

Cilj ovog dijela algoritma je stvoriti skup poligona koji su nastali rezanjem poligona P po gore konstruiranim tetivama, tj. skup poligona koji odgovaraju dijelovima particije. Unija ovih poligona je poligon P te među njima nema preklapanja. Algoritam se izvodi slijedno po produžecima rubova i produženim dijagonalama.

Algoritam 6 Konstrukcija dijelova particije

```
1:  procedure CONSTRUCTPARTITION(P, T)        ▷ T označava skup svih tetiva
2:      U ← {P}                               ▷ Inicijalizacija skupa poligona particije
3:      for all c ∈ T do
4:          for all p ∈ U do
5:              if intersects(c, p) then      ▷ Ako dužina c siječe poligon p
6:                  Podijeli p dužinom c na p_1 i p_2.
7:                  U ← U \ {p}
8:                  U ← U ∪ {p_1, p_2}
9:              end if
10:         end for
11:     end for
12: end procedure
```

Važno je primijetiti kako će u koraku 6 poligon p uvijek biti podijeljen na točno dva dijela. Ovo slijedi iz činjenice da se dužina c uvijek proteže između rubova poligona P, i činjenice da dijelovi $p \subset P$ ne mogu imati konkavnih kutova. Zato se dužina c, unutar poligona P može promatrati kao pravac.

5.2.3. Minimalni set cover greedy algoritam nad particijom

Minimalni set cover greedy algoritam nad ovako konstruiranom particijom poligona P izvodi se ekvivalentno greedy algoritmu nad triangulacijom, koji je detaljno opisan u kontekstu prethodnog algoritma.

5.3. Suboptimalnost rješenja

Prema teoremu D. S. Johnsona (Johnson, 1973), ako je c_{opt} optimalni broj čuvara, tada će ovaj algoritam za jednostavni poligon od n vrhova dati najviše $c_{opt} \log n$ čuvara. Ova granica proizlazi iz granice aproksimacijskih algoritama za rješavanje minimalnog set cover problema.

5.4. Analiza kompleksnosti prije optimizacije

Kompleksnost ovog algoritma uvelike ovisi o vrsti poligona koji mu je zadan. Ovdje je dana analiza najgoreg slučaja, koji bi se u praksi mogao postići tzv. poligonima velikih površina (engl. *large area polygon*), von Kochovim poligonima, i sl.

5.4.1. Broj tetiva

Broj tetiva[2] zbroj je broja dijagonala, broja produžetaka dijagonala te broja produžetaka rubova.

Kako se svaki vrh poligona teoretski može povezati s $n - 3$ drugih vrhova (kod, na primjer, konveksnih poligona), najgori slučaj za broj dijagonala iznosi $\mathcal{O}(n^2)$. Pod pretpostavkom da je moguće u ovom slučaju produljiti sve dijagonale i sve rubove na obje strane (što je, naravno, u praksi nemoguće), proizlazi da je broj produžetaka rubova $\mathcal{O}(2n)$, a broj produžetaka dijagonala $\mathcal{O}(2n^2)$. Zbog jednostavnosti, može se uzeti da je ukupni broj svih dužina koje dijele poligon $\mathcal{O}(n^2)$, jer su konstantni faktori i članovi manjeg stupnja zanemarivi.

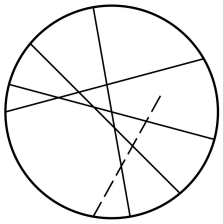
[2]Dužina koje dijele poligon, konstruiranih gore opisanim algoritmima.

5.4.2. Broj konveksnih komponenti

Za izvođenje broja konveksnih komponenti u najgorem slučaju koristi se teorem koji proizlazi iz Steinerovog problema o rezanju plohe (Michael, 2009):

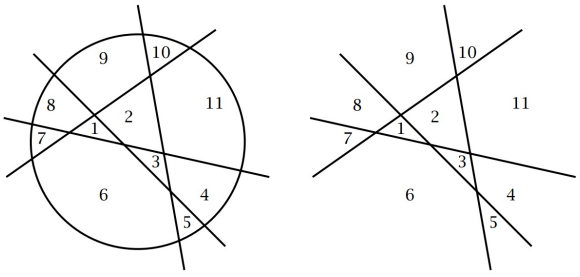
Teorem 3. *Maksimalan broj dijelova na koji se ploha može razrezati s k pravaca iznosi* $(k(k+1)/2)+1$.

Dokaz. Neka je $P(k)$ maksimalan broj dijelova na koji k pravaca može razrezati plohu. Pretpostavimo da je dana ploha razrezana na $P(k-1)$ dijelova uporabom $k-1$ pravaca. Dodaje li se k-ti pravac postepeno, kod svakog novog sjecišta s jednim od $k-1$ postojećih pravaca nastaje novi komad plohe (slika 5.3). Dodatno nastaje još jedan komad odrezan polupravcem koji počinje nakon zadnjeg sjecišta. Dakle, u k-tom koraku nastaje k novih komada. Dodatno, u nultom koraku vrijedi $P(0)=1$. Koristeći Gaussov poučak ili neku sličnu metodu, očigledno je $P(k)=(k(k+1)/2)+1$, što je i trebalo dokazati. □



Slika 5.3: Dodavanjem k-tog pravca stvara se k novih dijelova plohe. (Michael, 2009)

Kod particije poligona moguće je dužine ograničene rubovima poligona gledati kao pravce na tom dijelu plohe (slika 5.4). U najgorem slučaju, kada je broj tih dužina n^2 (kako je gore izvedeno), ukupni broj dijelova poligona bit će $n^2(n^2-1) \approx n^4$.



Slika 5.4: Dužine ograničene rubovima poligona mogu se promatrati kao pravci na tom dijelu plohe. (Michael, 2009)

5.4.3. Vremenska složenost

Konstrukcija produžetaka rubova izvodi se za svaki rub, pretragom svih ostalih pri traženju sjecišta, dakle u $\mathcal{O}(n^2)$. Konstrukcija dijagonala izvodi se za svaki par vrhova, pretragom svih bridova provjeravajući sijeku li dijagonalu ili ne, dakle u $\mathcal{O}(n^3)$.

Neka je m broj dijelova particije. Složenost konstrukcije particije iz zadanih dužina koje sijeku poligon je $\mathcal{O}(m)$.

Konstrukcija skupa S set cover problema (skupa skupova dijelova particije) izvodi se provjerom vidljivosti svakog pojedinog dijela particije iz svakog pojedinog vrha. Kako bi se izvela njezina složenost, uvodi se pojam *poligona vidljivosti*. Poligon vidljivosti $VP(P, p)$ je onaj dio poligona P koji je vidljiv iz točke p. Očigledno vrijedi $VP(P, p) \subset P$. Prema Ghosh (2010) i Lee (1983) poligon vidljivosti može se konstruirati u $\mathcal{O}(n)$. Dakle, vremenska složenost konstrukcija skupa S je $\mathcal{O}(nm)$.

Složenost greedy algoritma ovisi o broju dijelova particije m, te o broju vrhova n, pošto je to ujedno i broj skupova unutar S ($|S| = n$). Pretpostavimo da je kod stvaranja skupa S paralelno stvaran i skup S' u kojem su, za svaki dio particije zapisani vrhovi iz kojih je taj dio vidljiv. Nakon što je pronađen $s \in S$ koji u sebi sadrži najveći broj novih elemenata (dosad nepokrivenih dijelova particije), ovakva struktura omogućava uklanjanje tih dijelova iz svih ostalih elemenata skupa S koji ih sadrže bez potrebe za pretraživanjem, dakle u $\mathcal{O}(1)$ vremenu po dijelu, tj. u $\mathcal{O}(|s|)$ ukupno. Kako je u najgorem slučaju $|s| = m$, a broj čuvara n, vremenska složenost ovog dijela algoritma iznosi $\mathcal{O}(nm)$.

Na ovaj rezultat može se gledati i iz druge perspektive: kako je na početku izvođenja $|S| = n$, te $|s| = m$ (za najgori slučaj), te kako stoji činjenica da se svaki dio poligona iz svakog skupa unutar S uklanja točno jednom (pošto na kraju svi skupovi trebaju biti prazni jer ne postoji više nepokrivenih elemenata), očigledno je da algoritam izvodi nm uklanjanja, te mu je vremenska složenost $\mathcal{O}(nm)$. Za gore izvedeni $m = n^4$, vremenska složenost je $\mathcal{O}(n^5)$.

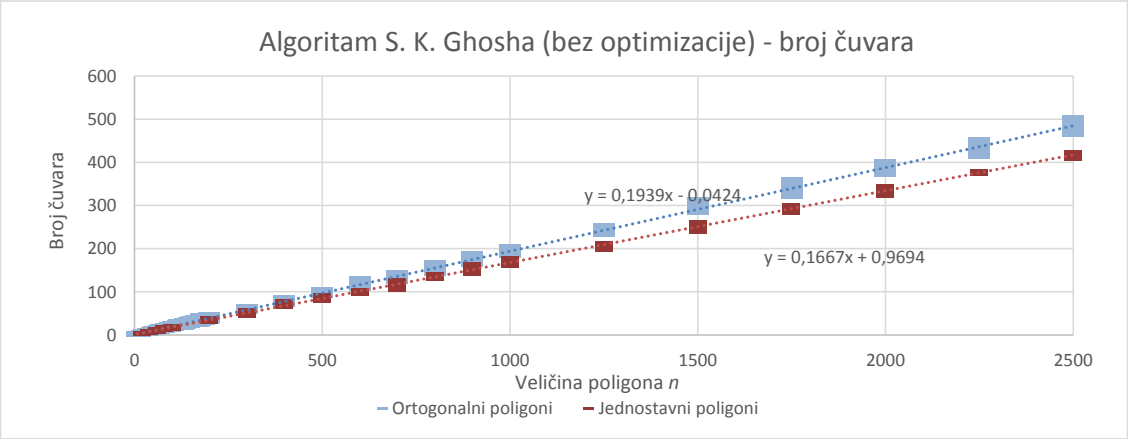
Kako je ovaj zadnji dio algoritma vremenski najsloženiji, ukupna vremenska složenost cijelog algoritma je $\mathcal{O}(n^5)$.

5.4.4. Prostorna složenost

Ekvivalentno vremenskoj složenosti, prostorno najsloženiji dio algoritma očigledno je skup S, čija je prostorna složenost $\mathcal{O}(mn) = \mathcal{O}(n^5)$. Dakle, prostorna složenost cijelog algoritma je $\mathcal{O}(n^5)$.

5.5. Rezultati testiranja prije optimizacije

5.5.1. Broj čuvara



Slika 5.5: Broj čuvara algoritma S. K. Ghosha prije optimizacije.

Kod slučajno generiranih razgranatih poligona linearni porast broja čuvara je očekivan. Kao i u prethodna dva algoritma, i ovdje jednostavni poligoni zahtijevaju nešto manje čuvara.

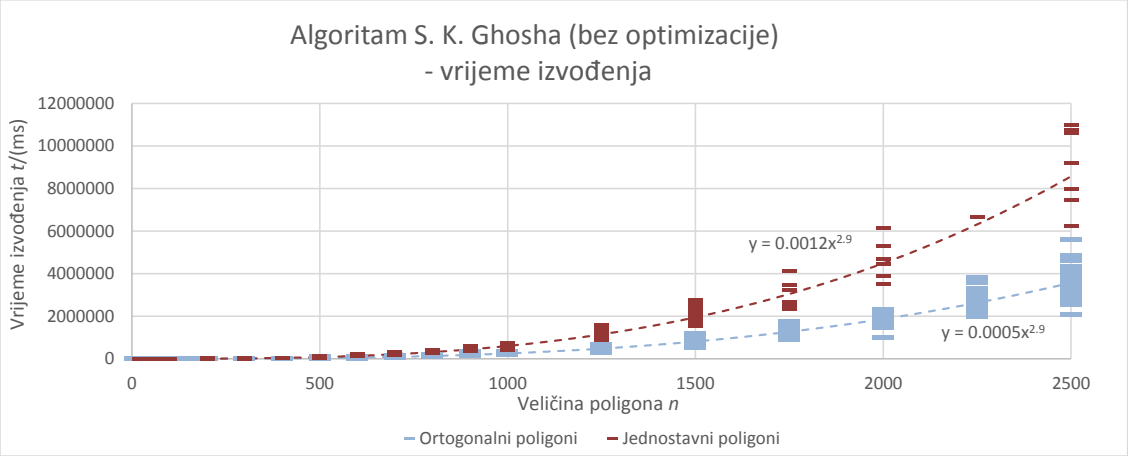
U usporedbi s algoritmom Avisa i Toussainta kod kojeg je koeficijent nagiba bio ≈ 0.327 te algoritmom minimalni set cover nad triangulacijom s koeficijentom ≈ 0.25, algoritam S. K. Ghosha daje očekivano bolje rezultate – koeficijent nagiba je ≈ 0.194 za ortogonalne, te ≈ 0.167 za jednostavne poligone.

Dakle, u prosjeku, algoritam daje ≈ 1.87 puta bolje rezultate od prvog, a ≈ 1.42 od drugog algoritma.

5.5.2. Vrijeme izvođenja

Iako je a priori izvedena vremenska složenost $\mathcal{O}(n^5)$, rezultati pokazuju bitno drugačije ponašanje algoritma. Kod obje vrste poligona a posteriori složenost je $\mathcal{O}(n^{2.9})$. Ipak, ovi rezultati ne čude uzmemo li u obzir podrazumijevani najgori slučaj kod a priori složenosti. U praksi je takav slučaj nemoguće postići za razgranate poligone na kakvima je algoritam testiran.

Također je zanimljivo usporediti dvije prikazane vrste poligona. Kako je broj unutarnjih dijagonala jednostavnih poligona u pravilu veći od ortogonalnih, veći je i broj tetiva, te je vrijeme izvođenja dulje.



Slika 5.6: Vrijeme izvođenja algoritma S. K. Ghosha prije optimizacije.

5.6. Optimizacija algoritma – smanjenje vremenske i prostorne složenosti

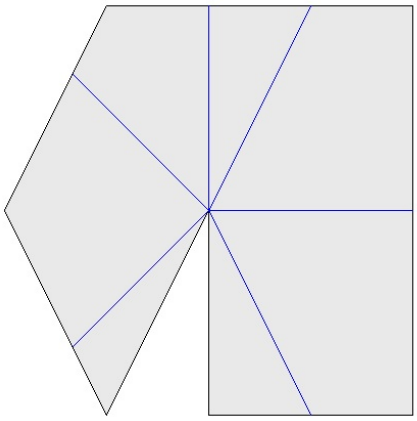
5.6.1. Opis optimizacije

Do sada opisani algoritam jednako je efikasan nad jednostavnim poligonima, kao i nad poligonima s rupama. Ograničavajući ga isključivo na jednostavne poligone, moguće je njegov rad značajno ubrzati. Tetive[3] su se do sada sastojale od produžetaka rubova, dijagonala te produžetaka dijagonala. Uvjet koji particija mora zadovoljavati, kao što je opisano u poglavlju 5.1, jest: dijelovi particije su ili cijeli vidljivi, ili cijeli nevidljivi iz nekog vrha poligona.

Drugim riječima, promatrajući jedan izolirani vrh poligona, tetive koje iz njega konstruiramo nužno moraju uključivati sve rubove njegovog poligona vidljivosti. Očigledno, unutarnje dijagonale konstruirane iz tog vrha ne spadaju u rubove poligona vidljivosti, zbog čega se postavlja pitanje: jesu li, prilikom konstrukcije particije, unutarnje dijagonale uopće značajne?

Uzimajući u obzir činjenicu da se dijelovi particije nikada neće protezati preko granica poligona vidljivosti nekog vrha, nego će biti ili cijeli unutar, ili cijeli izvan (ovo direktno proizlazi iz gore navedenog uvjeta), preostaje provjeriti je li moguće da, zanemarivanjem dijagonala, neka od preostalih tetiva ostane nepovezana s rubom poligona. Drugim riječima, je li moguće da krajevi neke tetive više nisu točke na rubovima poligona. Ovaj uvjet nužan je kako bi se osigurala konveksnost komponenti, kao što je

[3]Dužine koje su sjekle poligon te na temelju kojih je stvarana particija.



Slika 5.7: Particija produžecima rubova i dijagonala, bez samih dijagonala. Usporediti sa slikom 5.2.

komentirano u poglavlju 5.1. Detaljnom analizom načina na koji se ove dužine konstruiraju postaje jasno kako će uvjet uvijek biti zadovoljen. Naime, i produžeci rubova i produžeci dijagonala protežu se od nekog vrha poligona do prvog sjecišta pravca na kojem leže s nekim od rubova[4].

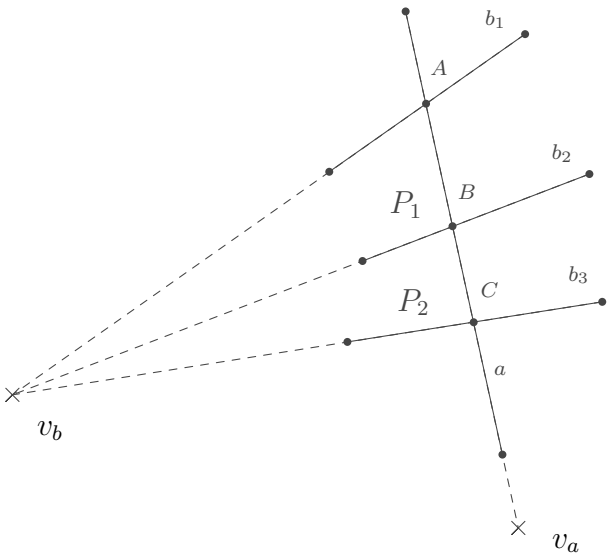
5.6.2. Vremenska složenost nakon optimizacije

Vremenska složenost ovog algoritma ovisi o broju vrhova n i broju dijelova particije m. Kako je broj vrhova nepromjenjiv, izvodi se samo novi m.

Broj tetiva, u najgorem slučaju, iznosi $\mathcal{O}(n^2)$, kako je prikazano u izvodu vremenske složenosti prije optimizacije. Ipak, nakon opisane optimizacije, postaje moguće ograničiti broj sjecišta nekog ruba poligona vidljivosti $VP(P, vi)$ s poligonom vidljivosti $VP(P, vj)$ na dva. U nastavku je ovo ograničenje složenosti detaljno opisano i potkrijepljeno dokazom.

Lema 3. *Za neka dva poligona vidljivosti $A = VP(P, v_i)$, $B = VP(P, v_j)$, nad istim jednostavnim poligonom P, vrijedi: neki rub poligona A može imati najviše dva sjecišta s poligonom B.*

[4]Za više detalja pogledati Bose et al. (2002)



Slika 5.8: Sjecišta dijagonale jednog poligona vidljivosti s tri dijagonale drugog poligona vidljivosti. Crtež uz dokaz leme 3.

Dokaz. Pretpostavimo suprotno: neka postoje poligoni vidljivosti $P_a = VP(P, v_a)$ te $P_b = VP(P, v_b)$ nad jednostavnim poligonom P za koje vrijedi: neki rub iz P_a siječe P_b u barem tri točke.

Neka pojam *novi rub* poligona vidljivosti označava one rubove koji leže na tetivama originalnog poligona. Drugim riječima, novi rubovi su oni rubovi koji čine unutarnje dijagonale originalnog poligona te nisu postojali prije konstrukcije poligona vidljivosti. Uz njih, poligoni vidljivosti sastoje se i od *starih rubova* koji leže na rubovima originalnog poligona.

Kako je očigledno da sjecišta promatrana u lemi 3 ne mogu biti na *starim* rubovima, pošto niti jedan rub poligona vidljivosti ne izlazi izvan poligona P, proizlazi da su tražena sjecišta nužno sjecišta između *novih* rubova[5].

Na slici 5.8 prikazane su središnje točke dvaju poligona vidljivosti v_a i v_b. Za poligon vidljivosti P_a iz točke v_a konstruiran je jedan novi rub a, dok su za poligon vidljivosti P_b iz točke v_b konstruirana tri nova ruba: b_1, b_2 i b_3 koji, po pretpostavci dokaza, sijeku rub a u tri točke, redom A, B i C.

Kako su dužine b_i rubovi poligona vidljivosti, očigledno je da za svaku od njih vrijedi da je unutrašnjost poligona P_b s točno jedne njihove strane. Drugim riječima, oni dijele dio ravnine neposredno oko sebe na dva dijela: poligon vidljivosti i ostatak

[5]Čak i kada ovo ne bi bio slučaj, lema 3 mogla bi se formulirati ovako: "*(...) vrijedi: neki novi rub poligona A može imati najviše dva sjecišta s novim rubovima poligona B*". Ovakav uvjet dovoljan je za kasnije izvođenje manjeg broja konveksnih komponenti, pošto njihov broj od $\mathcal{O}(n^4)$ drže baš sjecišta *novih* rubova kojih je $\mathcal{O}(n^2)$.

ravnine. Konkretno za dužinu b_2 slijedi da jedan od dijelova plohe P_1 ili P_2 mora sadržavati točke koje su izvan P_b. Neka je taj dio plohe, bez smanjenja općenitosti, P_2.

Tada slijedi, zbog prirode poligona vidljivosti, da se unutar trokuta CBv_b mora nalaziti barem jedna točka izvan originalnog poligona P. Kada bi vrijedilo suprotno, tj. kada bi cijeli trokut bio unutar poligona P, tada bi taj trokut cijeli nužno morao biti vidljiv iz v_b, što nije slučaj.

Kako vrijedi $VP(P, p) \subset P$, očigledno je da svi rubovi nekog poligona vidljivosti moraju biti unutar originalnog poligona nad kojim su nastali. Konkretno vrijedi: $\overline{CB}, \overline{Bv_b}, \overline{v_bC} \subset P$.

Kako ove stranice čine ciklus unutar poligona P, te kako unutar tog trokuta (ciklusa) postoje točke koje su izvan poligona P, slijedi da poligon P ima rupe, tj. da nije jednostavan poligon, što je u kontradikciji s početnom pretpostavkom. □

Uvodeći ovu optimizaciju, broj sjecišta između tetiva pada s $n^2 \cdot n^2 = n^4$ na $n \cdot n^2 = n^3$, zbog čega i broj dijelova particije pada s $m = n^4$ na $m = n^3$. Zbog toga i vremenska složenost cijelog algoritma pada na

$$\mathcal{O}(nm) = O(n^4) \tag{5.1}$$

5.6.3. Prostorna složenost nakon optimizacije

Prostorna složenost također ovisi o broju dijelova particije m te iznosi

$$\mathcal{O}(nm) = O(n^4) \tag{5.2}$$

5.7. Implementacija algoritma

5.7.1. Podatkovne strukture

Za dijelove particije je, kao i za cijeli poligon, korištena duplo ulančana lista vrhova. Koordinate vrhova predstavljene su decimalnim brojevima dvostruke preciznosti.

Skup S set cover dijela algoritma implementiran je *hash* tablicom, kao i svaki od njegovih elemenata, kako bi se brisanje pojedinih zapisa (dijelova particije) odvijalo u $\mathcal{O}(1)$.

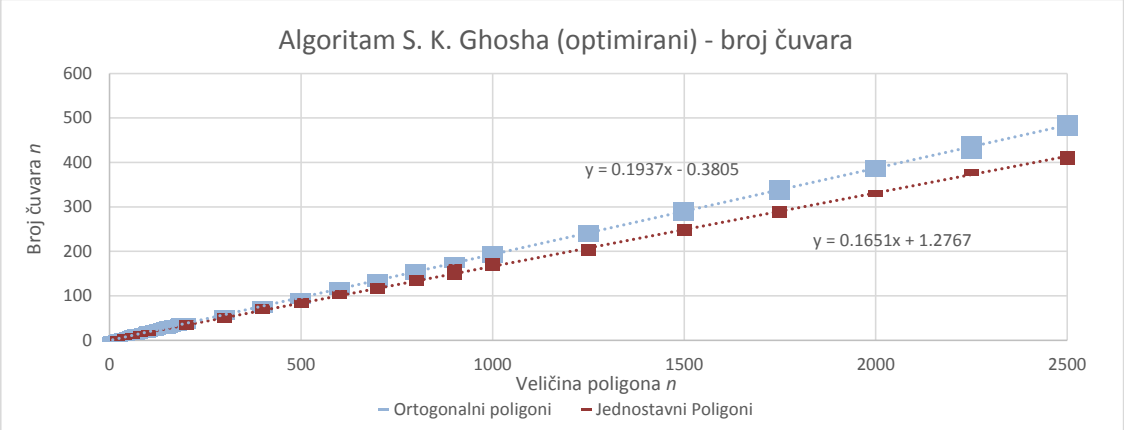
5.7.2. Implementacijski detalji

Najvažniji implementacijski detalj ovog algoritma odnosi se konkretno na programski jezik Javu i njezinu biblioteku za rad s 2D objektima. Na početku implementacije,

korištena je *java.awt.geom* paket (konkretno *Line2D* i *Point2D* razredi). Tijekom testiranja, nakon uočavanja niza anomalija u njegovom ponašanju, paket je zamijenjen vlastitom implementacijom potrebnih geometrijskih funkcija: funkcije za provjeru sijeku li se dvije dužine, dužina i pravac ili dva pravca, funkcija za određivanje tog sjecišta, funkcije za provjeru nalazi li se zadana točka na zadanoj dužini ili pravcu te nalazi li se unutar zadanog poligona. Također, korištene su i geometrijske funkcije signedArea i clockwise opisane u kontekstu algoritma Avisa i Toussainta.

5.8. Rezultati testiranja nakon optimizacije

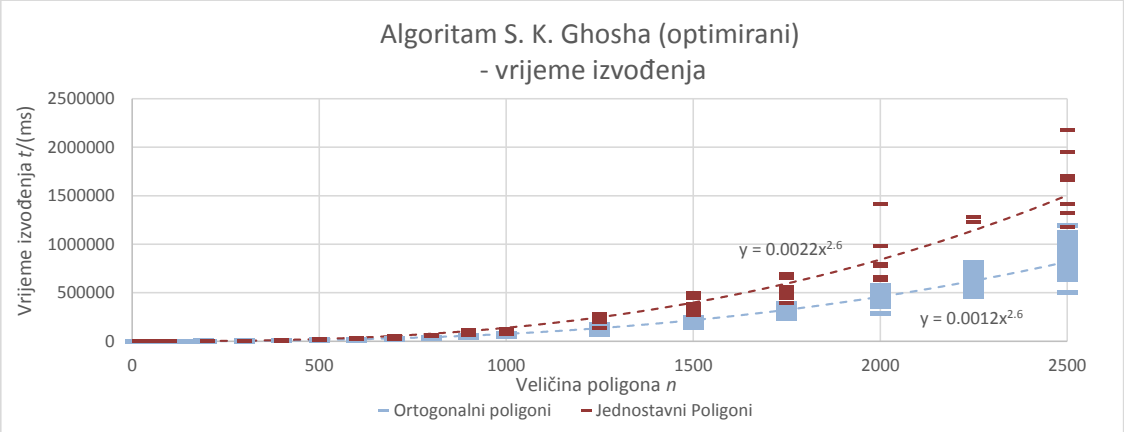
5.8.1. Broj čuvara



Slika 5.9: Broj čuvara algoritma S. K. Ghosha nakon optimizacije.

Kao što je i bilo predviđeno, broj pronađenih čuvara nakon optimizacije ostaje isti, jer kvaliteta rješenja ovog algoritma ovisi isključivo o kvaliteti rješenja set cover problema. Pošto je za njegovo rješavanje korišten isti algoritam kao i prije optimizacije, rezultati su jednako udaljeni od optimalnih.

5.8.2. Vrijeme izvođenja



Slika 5.10: Vrijeme izvođenja algoritma S. K. Ghosha nakon optimizacije.

A priori analiza daje smanjenje složenosti nakon optimizaicije s $\mathcal{O}(n^5)$ na $\mathcal{O}(n^4)$. Prije optimizacije, rezultati testiranja dali su vremensku složenost od $\mathcal{O}(n^{2.9})$. Kao što bi i bilo za očekivati, ova složenost razmjerno pada nakon implementirane optimizacije na $\mathcal{O}(n^{2.6})$.

Dodatno, zanimljivo je primijetiti kako su se nakon optimizacije jednostavni poligoni približili ortogonalnima. Ova pojava može se objasniti većim brojem uutarnjih dijagonala jednostavnih poligona koje su optimizacijom sve uklonjene.

6. Buduća istraživanja

Složenost opisanog modernog algoritma od $\mathcal{O}(nm)$ drže dva njegova dijela: konstrukcija skupa S te razrješavanje set cover problema greedy algoritmom. Optimizacija nekog od ta dva dijela ne bi umanjila ukupnu kompleksnost, ali bi uvelike ubrzala izvođenje algoritma. U nastavku je teorijski opisana optimizacija stvaranja skupa S.

6.1. Optimizacija konstrukcije set cover problema nad konveksnim komponentama

Prilikom konstrukcije skupa S, najveći dio procesorskog vremena koristi se za računanje geometrijskih funkcija: traženje sjecišta između linija, ispitivanje nalazi li se neka točka unutar poligona, siječe li neka dužina poligon i sl. Sve te operacije procesorski su iznimno zahtjevne. Ova optimizacija zaobilazi veliku većinu geometrijskih operacija prilikom stvaranja skupa S koristeći dualni graf particije[1] koji se, paralelno sa skupom S, konstruira tijekom particioniranja.

Optimizacija se najbolje prikazuje u dva stupnja. Prvi stupanj optimizacije stvara skup S paralelno s particijom, dok drugi stupanj zaobilazi većinu geometrijskih operacija prilikom stvaranja skupa S koristeći dualni graf particije.

6.1.1. Paralelna konstrukcija particije i set cover problema

Dosadašnja konstrukcija skupa S zahtijevala je provjeru vidljivosti između svakog para konveksne komponente i vrha poligona. Ako je za određivanje vidljivosti potrebno k vremena, tada je ukupno vrijeme konstrukcije $k \cdot nm$, gdje je n broj vrhova poligona, a m broj konveksnih komponenti.

Neka je implementirana nova podatkovna struktura koja proširuje konveksnu komponentu na način da, uz sam poligon komponente, sprema i listu vrhova iz kojih je

[1] Dualni graf particije ekvivalentan je dualnom grafu triangulacije opisanom u poglavlju 4.1.2.

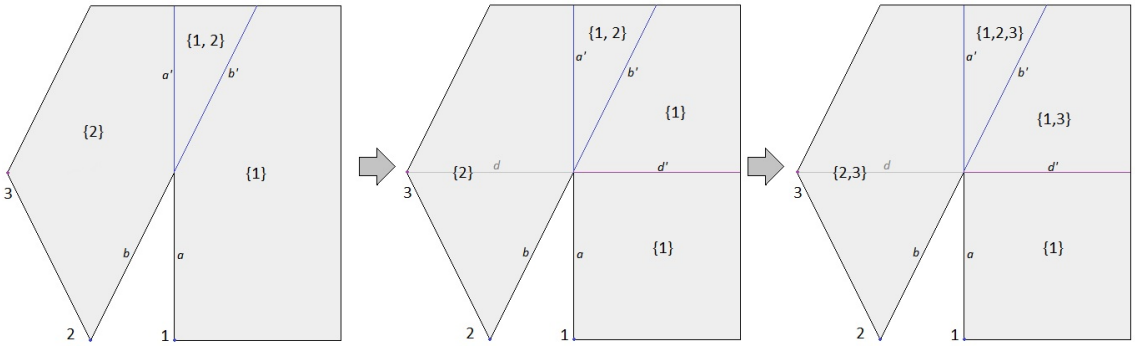
komponenta vidljiva. Očigledno je kako se ovakva struktura može koristiti za direktnu konstrukciju skupa S. Također, ona se može koristiti i za direktan pristup informaciji "koje su sve konveksne komponente vidljive iz nekog vrha"[2].

Za razliku od dosadašnjeg tijeka algoritma, koji je prolazio svim konstruiranim tetivama te svakom od njih *rezao* [3] sve dosadašnje konveksne komponente kroz koje prolazi, optimirani algoritam prolazi tetivama grupirajući ih po vrhovima iz kojih su nastale. Drugim riječima, za vrh v_i se uzimaju sve tetive koje su nastale kao rubovi poligona vidljivosti $VP(P, v_i)$ te se njima dijele sve dosad konstruirane konveksne komponente.

Komponente p_1 i p_2 nastale rezanjem komponente p prirodno nasljeđuju njezinu listu vidljivih vrhova.

Nakon ovakve podjele poligona svim tetivama nastalima iz vrha v_i, poligon P podijeljen je na dvije skupine konveksnih komponenti: one cijele vidljive te one cijele nevidljive iz v_i. Algoritam sada prolazi svim konveksnim komponentama te svakoj vidljivoj iz v_i dodaje i (ili neki drugi identifikator vrha) na listu vidljivih vrhova te komponente.

Na ovaj način broj provjera vidljivosti je prepolovljen, pošto se za svaki vrh provjerava vidljivost komponenti particije nastale tetivama iz tog i njemu prethodnih vrhova, dok se kod neoptimiranog algoritma vidljivost uvijek provjeravala za komponente potpune particije.



Slika 6.1: Podjela poligona tetivom d' iz vrha 3. Skupovi vidljivih vrhova konveksnih komponenti zapisani su u vitičastim zagradama.

[2]Kada bi sve konveksne komponente bile spremljene kao stupci 2-dimenzionalne tablice u kojoj bi retci bili pojedini vrhovi poligona, a elementi tablice *boolean* vrijednosti *true* ili *false* koje označavaju je li pojedina komponenta vidljiva iz pojedinog vrha, tada bi takva tablica bila ekvivalentna tabličnom zapisu skupa S, što bi omogućilo zaobilaženje dodatnih transformacija ove nove podatkovne strukture u skup S.

[3]Podjela konveksnih komponenti na dva dijela opisana je u poglavlju 5.2.2.

6.1.2. Optimizacija dualnim grafom particije

Neka je podatkovna struktura konveksne komponente proširena još jednom listom: listom susjednih konveksnih komponenti. Na ovaj način na konveksne komponente može se gledati kao na vrhove dualnog grafa, ekvivalentno dualnom grafu triangulacije. Važno je napomenuti kako u ovom slučaju graf ipak nije nužno stablo.

Propozicija 1. *Poligon vidljivosti je jednostavan poligon.*

Iz ovoga slijedi da je svaki par točaka u poligonu vidljivosti povezan nekom stazom koja je cijela unutar poligona[4].

Ako postoji dualni graf konveksnih komponenti, tj. ako postoji informacija o *susjednosti* dvaju konveksnih komponenti, tada je moguće, koristeći *BFS* ili neku sličnu strategiju širenja, obići sve konveksne komponente unutar poligona vidljivosti. Granice širenja bile bi tetive trenutno promatranog vrha.

Drugim riječima, neka je v_i trenutno promatrani vrh, a T skup tetiva koje su iz tog vrha konstruirane. Neka je skup konveksnih komponenti U već podijeljen tetivama iz T, te neka je dualni graf ograničen tim tetivama[5]. Sada je potrebno pronaći samo jednu komponentu koja je vidljiva iz v_i, nakon čega se iz nje informacija o vidljivosti proširi ostatkom dualnog grafa (ograničenog tetivama).
Za tu prvu komponentu može se uzeti neka koja dodiruje vrh v_i. Dodatnim podatkovnim strukturama moguće je osigurati ovakav pronalazak u $\mathcal{O}(1)$.

Ukupno ubrzanje ovom optimizacijom ovisi o razlici između vremena određivanja vidljivosti dviju točaka i vremena širenja dualnim grafom. Kod jednostavnih implementacija provjere vidljivosti koje imaju $\mathcal{O}(n)$ kompleksnost, pod pretpostavkom da se širenje s jednog vrha grafa na drugi može obaviti u $\mathcal{O}(1)$, ova optimizacija značajno smanjuje vremensku složenost prebacivanja problema umjetničke galerije u set cover problem.

S druge strane, čak i korištenjem naprednih algoritama za provjeru vidljivosti, ova optimizacija predstavlja značajno ubrzanje.

[4]Trivijalan slučaj za takvu stazu bi bila staza koja iz prve točke ide ravno do središnje točke poligona vidljivosti pa zatim ravno do druge točke. Ovo je uvijek moguće jer je poligon vidljivosti zvjezdasti poligon.

[5]Ovaj mehanizam jednostavno se može izvesti na sljedeći način: nakon podjele komponenti tetivama iz T, potrebno je osvježiti vrhove dualnog grafa susjednih komponenti. Ovo osvježavanje može se izvesti u dva koraka. Prvi korak spaja nove dijelove komponenti sa svim susjednima, osim s onima s kojima su u prethodnom koraku činile istu komponentu (te ih sada dijeli tetiva). Nakon izvršenog obilaska poligona vidljivosti, u drugom koraku se spajaju i ovi parovi komponenti.

7. Zaključak

Problem umjetničke galerije će bez sumnje još dugo vremena biti opsežno proučavan kao problem čiji su algoritmi za rješavanje upotrebljivi u nekolicini raznovrsnih područja: računalnoj geometriji, računalnoj grafici, automatici, itd. Kako se radi o NP-teškom problemu, u praksi se koriste isključivo aproksimacijski algoritmi, koji su zbog toga posebno zanimljivi.

Algoritam S. K. Ghosha, detaljno opisan u ovom radu, jedan je od najnovijih neiteracijskih aproksimacijskih algoritama problema umjetničke galerije. Pretvarajući problem u tzv. minimalni set cover problem konstruiranjem particije u tzv. konveksne komponente nad zadanim poligonom, algoritam prenosi problem u opsežno istraženo područje teorije skupova, koristeći tako niz već postojećih metoda za razrješavanje problema. Rubovi konstruirane particije su produžeci unutarnjih dijagonala i rubova poligona. U ovakvom obliku, koristeći najjednostavniju pohlepnu heuristiku za rješavanje dobivenog set cover problema, složenost algoritma je $\mathcal{O}(n^5)$.

Uvođenjem optimizacije prilikom koje se kod konstrukcije particije zanemaruju same unutarnje dijagonale, te se uzimaju u obzir samo produžeci tih dijagonala i rubova poligona, složenost algoritma pada na $\mathcal{O}(n^4)$, što je i dokazano.

Testiranje implementacije ovog algoritma izvršeno je na dvije skupine poligona: 10 000 primjera ortogonalnih, te 1000 primjera jednostavnih poligona, čije se veličine kreću od 10 do 2500 vrhova. Rezultati testiranja zanimljivi su iz nekoliko razloga. Kao prvo, oni ukazuju na značajnu razliku između najgorih slučaja kakvi su promatrani kod a priori računanja složenosti i realnih primjera na kakvima je algoritam testiran. Prije optimizacije, složenost algoritma na realnim primjerima je oko $\mathcal{O}(n^{2.9})$, dok nakon optimizacije ona pada na $\mathcal{O}(n^{2.6})$. Nadalje, testni rezultati pokazuju i razliku između jednostavnih i ortogonalnih poligona u kontekstu ovog problema. Naime, jednostavni poligoni u pravilu imaju više unutarnjih dijagonala od ortogonalnih poligona iste veličine, zbog čega trebaju oko 2.4 puta više vremena za razrješavanje. Nakon optimizacije, tj. nakon uklanjanja unutarnjih dijagonala iz skupa rubova particije poligona, ovaj faktor pada na oko 1.8, jer je sada broj tetiva među ovim vrstama poligona sličniji.

LITERATURA

David Avis i Godfried T Toussaint. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recognition*, 13(6):395–398, 1981.

Prosenjit Bose, Anna Lubiw, i J Ian Munro. Efficient visibility queries in simple polygons. *Computational Geometry*, 23(3):313–335, 2002.

Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

Marcelo C. Couto, Pedro J. de Rezende, i Cid C. de Souza. Instances for the Art Gallery Problem, 2009. *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

Steve Fisk. A short proof of chvátal's watchman theorem. Journal of Combinatorial Theory, Series B, *24(3):374, 1978.*

Michael R Garey, David S Johnson, Franco P Preparata, i Robert E Tarjan. Triangulating a simple polygon. Information Processing Letters, *7(4):175–179, 1978.*

Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons. Discrete Applied Mathematics, *158(6):718–722, 2010.*

Héctor González-Baños. A randomized art-gallery algorithm for sensor placement. U Proceedings of the seventeenth annual symposium on Computational geometry, *stranice 232–240. ACM, 2001.*

David S Johnson. Approximation algorithms for combinatorial problems. U Proceedings of the fifth annual ACM symposium on Theory of computing, *stranice 38–49. ACM, 1973.*

DT Lee. Visibility of a simple polygon. Computer Vision, Graphics, and Image Processing, *22(2):207–221, 1983.*

T. S. Michael. How to Guard an Art Gallery and Other Discrete Mathematical Adventures. *Johns Hopkins University Press, 2009.*

Joseph O'Rourke. Art gallery theorems and algorithms. *Oxford University Press, 1987.*

Robert E Tarjan i Christopher J Van Wyk. An o(n\log\logn)-time algorithm for triangulating a simple polygon. SIAM Journal on Computing, *17(1):143–178, 1988.*

Wikipedia. Art gallery problem — wikipedia, the free encyclopedia, 2014. URL `http://en.wikipedia.org/w/index.php?title=Art_gallery_problem&oldid=614497808`. *[Online; accessed 31-May-2015].*

Algoritmi za problem umjetničke galerije

Sažetak

Problem umjetničke galerije je matematički problem određivanja položaja najmanjeg broja čuvara koji svojim vidnim poljima pokrivaju cijeli zadani tlocrt opisan poligonom. U ovom radu promatrani su isključivo čuvari na vrhovima jednostavnih poligona. Cilj rada je dati uvod u moderni algoritam S. K. Ghosha kroz dva jednostavnija algoritma te ga detaljno opisati. Zatim opisati optimizaciju koja vremensku i prostornu složenost spušta s $\mathcal{O}(n^5)$ na $\mathcal{O}(n^4)$. Dani rezultati testiranja opravdavaju ovu optimizaciju, pokazuju veliku razliku između najgoreg i prosječnog praktičnog slučaja problema te ukazuju na bitne razlike između slučajnih jednostavnih i ortogonalnih poligona. Na kraju je isključivo teorijski opisana dodatna optimizacija algoritma, koja mu ne smanjuje složenost, ali poboljšava performanse.

Ključne riječi: Problem umjetničke galerije, Chvátalov teorem umjetničke galerije, algoritam Avisa i Toussainta, algoritam S. K. Ghosha, set cover problem, pohlepna heuristika.

Art Gallery Problem Algorithms

Abstract

The Art Gallery problem deals with finding minimal number of guarding positions so as to enable the guards to see every point of a given polygon. Focusing on the problem of vertex guards in simple polygons, this thesis offers a comprehensive introduction through two simple algorithms, a description and complexities calculation of a modern algorithm by S.K. Ghosh, along with an optimization that decreases space and time complexity from $\mathcal{O}(n^5)$ to $\mathcal{O}(n^4)$. The comparison of the optimization values to a set of test results indicates an important distinction between worst-case and real-life scenarios and some differences between the orthogonal and the simple polygons. Finally, there is a theoretical description of an alternative optimization that should not change the algorithm's complexity but could improve its performance significantly.

Keywords: Art gallery problem, Chvátals art gallery theorem, Avis – Toussaint algorithm, Ghosh's algorithm, set cover problem, greedy heuristics.