SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 538

Optimizacija 3D modela smanjenjem broja poligona uz pomoć evolucijskih algoritama

Filip Defar

Zagreb, lipanj 2013.

Sadržaj

1 Uvod

[1] Mreže poligona su se postavile kao standard za prikazivanje 3d modela u brojnim područjima računalne grafike. Njihova veličina i kompleksnost raste brže od mogućnosti grafičkog sklopovlja pa se tako javlja potreba za pojednostavljivanjem kompleksnih modela kako bi se poboljšale performanse aplikacija. Ovaj postupak ima posebnu važnost u interaktivnim računalnim aplikacijama koje su namijenjene za izvedbu na kućnim računalima. Ipak čak i vrlo napredno sklopovlje nailazi na probleme prilikom obrade izuzetno kompleksnih modela, poput onih dobivenih 3d skeniranjem stvarnih objekata. Ručno pojednostavljivanje objekata je iznimno zamoran, spor i tegoban posao pa se teži razvoju sustava za automatsko uklanjanje poligona iz kompleksnih modela. Osnova problema je odabrati i ukloniti poligone koji će minimalno deformirati objekt, odnosno od originalnog modela dobiti pojednostavljeni s manjim brojem poligona, ali što vjerniji originalu. Unutar ovog rada će se razmatrati korištenje evolucijskih algoritama, primarno genetskih algoritama i genetskog programiranja, na ovom problemu.

U drugom poglavlju se opisuje općenit problem optimizacije 3d modela te postojeće metode za rješavanje tog problema. U trećem poglavlju se vrlo sažeto opisuju evolucijski algoritmi. Centralni dio rada se nalazi u četvrtom poglavlju u kojem se opisuju različite inačice genetskog algoritma kao i dobiveni rezultati. Unutar petog poglavlja nalazi se opis pokušaja rješavanja problema genetskim programiranjem i dobiveni rezultati.

2 Problem optimizacije modela virtualnih likova

2.1 Osnovni način zapisa modela

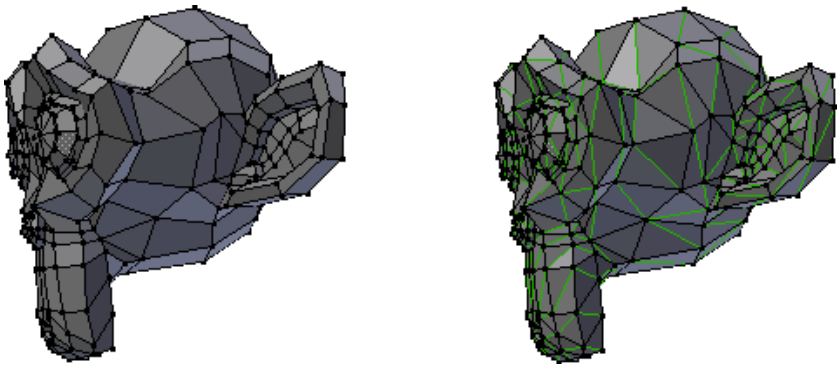
Osnovan problem koji je potrebno riješiti kako bi omogućili prikaz 3d računalne grafike je spremanje podataka o objektima koje želimo prikazati. Kako je potreban način zapisivanja koji je jednostavan i pogodan za obradu na računalu, standardno se model iz realnog svijeta razlaže na skup međusobno povezanih poligona [4]. Iako mnogi zapisi podržavaju poligone s proizvoljnim brojem stranica, korištenje isključivo trokuta ne predstavlja nikakvo ograničenje jer se svaki poligon s N stranica može prikazati i kao N-2 međusobno povezanih trokuta (slika 1).

Najjednostavniji zapis modela sastoji se od popisa svih točaka koje se nalaze u modelu zajedno s informacijom kako ih spojiti u poligone. Zapis svake točke se sastoji od X, Y i Z koordinate koje predstavljaju položaj te točke prostoru. Zapis svakog poligona se sastoji od popisa točaka koje se nalaze u njemu, a kako te točke čine vrhove poligona one se često nazivaju vrhovi (eng. vertex). Često se vrhovi poligona upisuju u dogovorenom redoslijedu, npr. smjeru suprotnom od kazaljke na satu, što omogućava izbjegavanje eksplicitnog zapisivanja smjera normale već se on može izračunati. Uz navedene osnovne elemente 3d modela (vrh i poligon) potrebno je još uvesti i pojam ruba (eng. edge). Rub je dio poligona, odnosno jedna njegova stranica i povezuje dva različita vrha. Podatke o rubovima nije nužno posebno zapisivati jer poligoni unutar sebe sadržavaju informacije o njima.

Ovaj zapis je dovoljan za opisivanje strukture statičkih 3d modela proizvoljne kompleksnosti, a moguće ga je i dodatno proširiti kako bi sadržavao dodatne informacije potrebne za realističan prikaz 3d modela, npr. informacije o teksturama i mapama izbočina. Kako ova proširenja nisu ključna za problematiku rada nećemo ulaziti u detalje njihove implementacije.

2.2 Mreža polu-rubova

Mreža polu-rubova (eng. half-edge mesh) za razliku od prethodno opisanog načina zapisa se primarno oslanja na spremanje podatka o povezanosti rubova s ostalim elementima modela. Svaki rub ima informacije o dva vrha koje sadržava, poligonima kojima pripada i dva susjedna ruba u tim poligonima. Ako rub podijelimo u dva dijela, tako



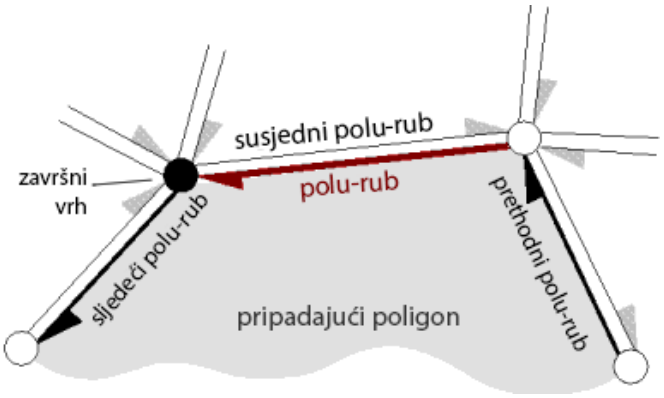
Slika 1: rastavljanje modela na trokute

da dobijemo dva elementa koje možemo nazvati polu-rubovi, koji su usmjereni i svaki sadrži informacije o jednom poligonu koji mu pripada kao i o susjednom polu-rubu u tom poligonu, dobivamo mrežu polu-rubova. Također svaki polu-rub ima informacije o svom susjedu, koji mora biti suprotne orijentacije. Ako rub sadrži dva vrha A i B, tada je nužno jedan njegov polu-rub usmjeren od A prema B, a drugi od B prema A. Slijedi popis referenci koji pojedini elementi modela sadrže.

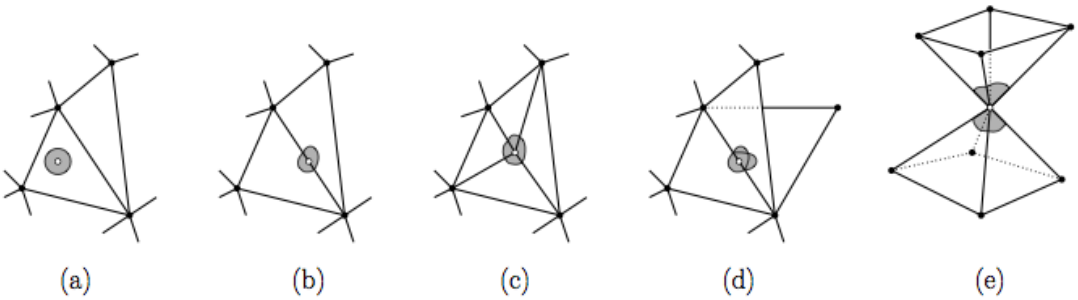
- Svaki vrh sadrži referencu na jedan polu-rub koji iz njega izlazi
- Svaki poligon sadrži referencu na jedan polu-rub kojeg sadrži
- Svaki polu-rub sadrži referencu na (slika 2):
  - završni vrh tog polu-ruba
  - poligon kojem pripada
  - sljedeći polu-rub unutar poligona kojem pripada
  - prethodni polu-rub unutar poligona (nije nužno, ali može biti korisno)
  - susjedni polu-rub

Korištenjem ovih veza moguće je prolaziti kroz model na razne načine. Na primjer budući da svaki polu-rub sadrži informaciju o sljedećem rubu unutar poligona, a svaki poligon sadrži informaciju o jednom polu-rubu unutar njega, moguće je prolaziti svim polu-rubovima nekog poligona.

Budući da su dva susjedna polu-ruba uvijek suprotne orijentacije moguće je lako prolaziti kroz sve polu-rubove koji izlaze iz nekog vrha, dovoljno je krenuti iz bilo kojeg polu-ruba koji izlazi iz zadanog vrha, prijeći na njemu susjedan polu rub (koji je nužno



Slika 2: isječak mreže polu rubova



Slika 3: Primjeri ispravnih (a, b, c) i neispravnih (d, e) stanja modela, preuzeto iz [7]

suprotne orijentacije pa je tako zadani vrh njemu završan), te zatim prijeći na sljedeći rub unutar tog poligona. Ovim koracima smo prešli na sljedeći polu-rub koji izlazi iz tog vrha, te ih je moguće ponavljati sve dok ne dođemo ponovo do početnog polu-ruba.

Ovaj zapis je također pogodan za modifikaciju modela, pa se iz tog razloga često koristi prilikom pojednostavljivanja modela.

Iako je zapis mreže polu-rubova vrlo fleksibilan, on ima i svoja ograničenja. Kako bi zapis bio valjan svaki rub mora sadržavati točno 2 polu-ruba, odnosno biti povezan s točno 2 poligona. Također svi polu-rubovi koji izlaze iz nekog vrha moraju pripadati poligonima koji su međusobno povezani rubovima, iako ta veza ne mora biti neposredna. Drugim riječima, bliska okolina svake točke na površini modela mora činiti topološki disk [7]. Ovaj zahtjev je prirodan i ispunjava ga većina postojećih modela. Ipak važno je napomenuti da postupcima pojednostavljivanja modela ovaj uvjet može biti prekršen što narušava integritet mreže polu-rubova pa je na to potrebno obratiti pažnju. Na slici 3 se vide 3 primjera zadovoljavanja tog uvjeta (a, b, c) i 2 primjera njegovog kršenja (d,e).

2.3 Motivacija za automatsko uklanjanje trokuta

Prilikom modeliranja objekata najčešće se stvaraju modeli s velikim brojem poligona (high-polygon model) a zatim se naknadno pretvaraju u pojednostavljeni model. Ovakav postupak se primjenjuje zato što je 3d umjetnicima lakše oblikovati model koji ima visok broj poligona jer se na njih, umjesto direktnog crtanja trokuta, mogu primjenjivati apstraktnije metode modeliranja koje podsjećaju na rad s glinom, a uz to vrlo često se detaljniji model koristi za preslikavanje informacija o normalama na jednostavniji objekt što omogućava realistični prikaz detalja bez korištenja dodatnih poligona. Ovaj postupak je poznat kao preslikavanje neravnina (eng. bump mapping). Potreba za pojednostavljivanjem kompleksnih modela se pojavljuje i prilikom 3d skeniranja objekata, jer taj postupak generira modele s vrlo velikim broj poligona od kojih mnogi nastaju zbog šuma.

Centralni problem optimizacije modela uklanjanjem poligona je određivanje utjecaja uklanjanja pojedinog poligona na deformaciju modela. U pravilu je poželjno uklanjati poligone iz najmanje zakrivljenih dijelova modela jer će ti koraci najmanje deformirati površinu modela [1]. Ipak, potrebno je uzeti i druge aspekte u obzir kao što su veličina trokuta kojeg uklanjamo i hoće li uklanjanje nekog trokuta uzrokovati preklapanje ostalih trokuta ili ugroziti valjanost modela.

Pojednostavljivanje modela moguće je raditi ručno, ali taj postupak je izrazito dugotrajan i mukotrpan, pogotovo kod modela s izrazito visokim brojem trokuta, kao što su oni dobiveni 3d skeniranjem. Iako postoji velik broj postupaka automatskog pojednostavljivanja modela, zbog njegove kompleksnosti ovaj problem je i dalje otvoren i traže se nova, efikasnija rješenja.

2.3.1 Spajanje koplanarnih poligona

U ovom postupku se traže poligoni koji su koplanarni ili približno koplanarni te se spajaju u veće poligone [3]. Dobiveni poligoni se zatim rastavljaju u trokute, čiji bi broj trebao biti manji od početnog broja trokuta. Ovu metodu je moguće proširiti s algoritmom koji traži optimalni način rastavljanja spojenih poligona na trokute.

2.3.2 Kontrolirano uklanjanje rubova ili vrhova

Ova metoda iterativno uklanja komponente modela (rubove ili vrhove) koji se biraju pomoću funkcije dobrote [3]. Za funkciju dobrote se najčešće koristi procjena zakrivljenosti površine. Početna verzija algoritma uklanja vrhove koji prolaze lokalne geometrijske i topološke provjere, te se nastale rupe zatim popunjavaju novim trokutima. Uklanjanje vrhova je moguće zamijeniti urušavanjem rubova, što uklanja potrebu za popunjavanjem rupa te tako pojednostavljuje postupak. U algoritam je moguće uvesti računanje globalne pogreške jer uzastopno uklanjanje komponenata s malom lokalnom pogreškom može rezultirati velikom globalnom pogreškom. Za računanje globalne pogreške postoje razne egzaktne metode kao i metode aproksimacije. Također postoje varijante algoritma koje naknadno pokušavaju optimizirati površinu korištenjem postupka okretanja rubova.

2.3.3 Optimizacija energijske funkcije

U ovom postupku definirana je energijska funkcija koja mjeri kvalitetu pojednostavljenog modela [3]. Algoritam iterativno obavlja urušavanje ruba, okretanje ruba ili lomljenje ruba tako da odabrani potez minimalno povećava vrijednost energijske funkcije.

2.3.4 Grupiranje vrhova

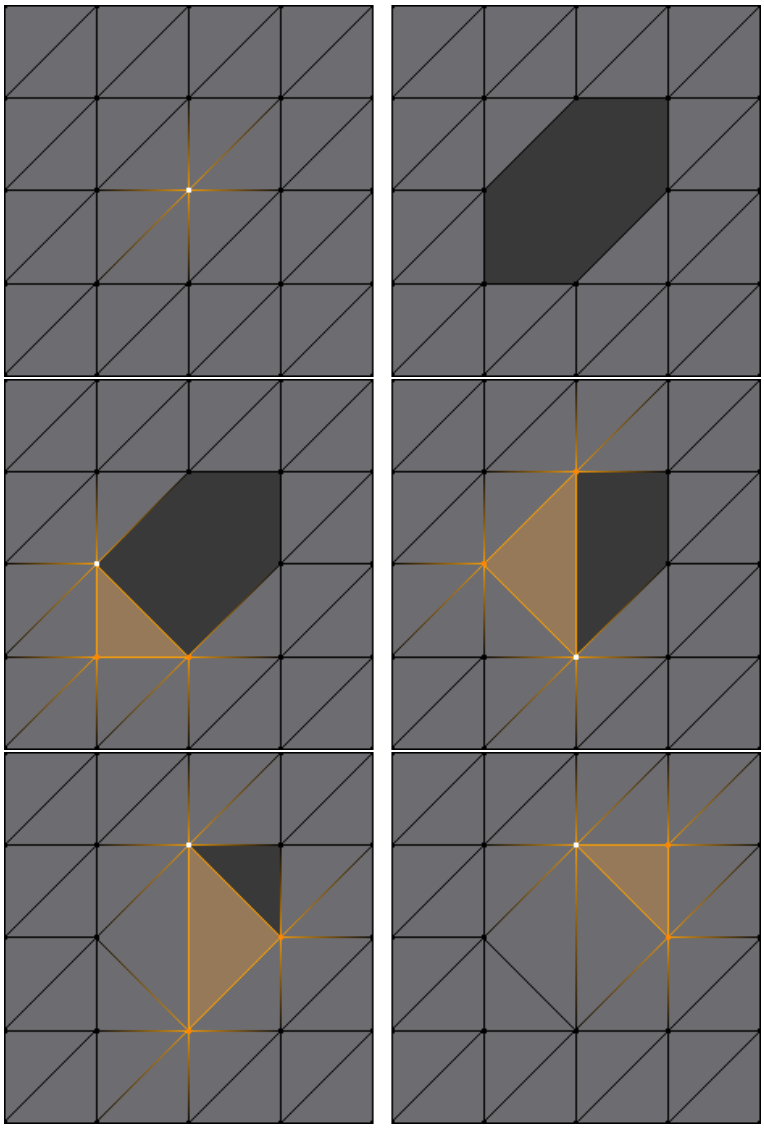
Algoritam grupira vrhove koji su međusobno blizu, te za svaku grupu vrhova određuje novi vrh s kojim će ih zamijeniti [3]. Ova metoda je učinkovita ali nema garancije kako će se sačuvati detalji modela ili čak njegov osnovan oblik.

2.4 Postupci modifikacije površine modela

2.4.1 Uklanjanje vrha

Postupak uklanjanja vrha se odvija u dva koraka. U prvom koraku se odabrani vrh uklanja zajedno s N trokuta koji su sadržavali taj vrh [1]. Zatim se nastala rupa u modelu popunjava novim trokutima. Za potpuno zatvaranje nastale rupe dovoljno je iskoristiti N-2 trokuta, pa tako ovim postupkom za svaki uklonjeni vrh gubimo 2

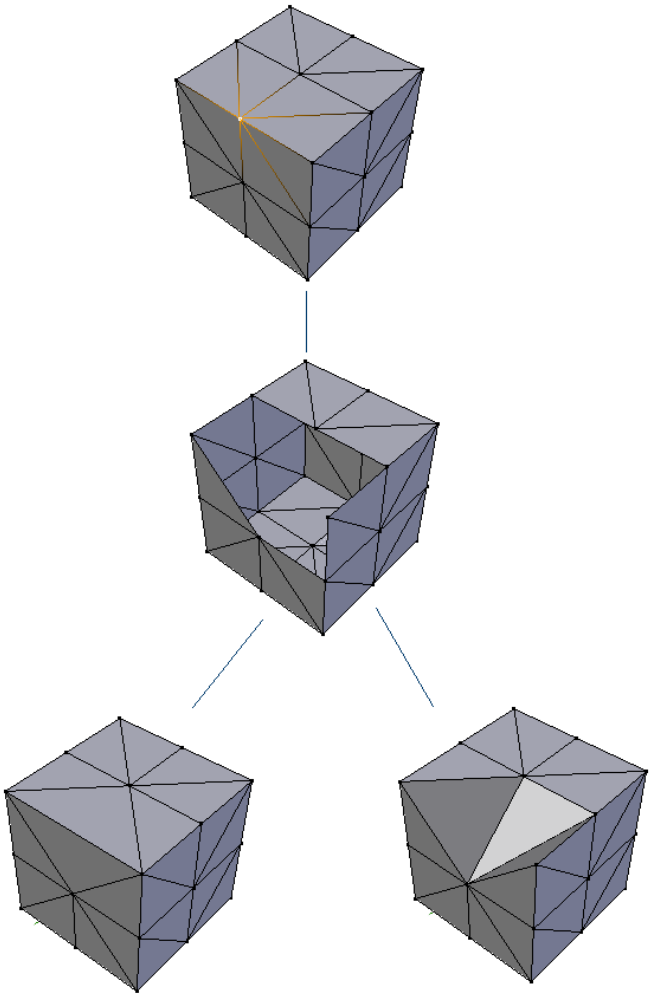
trokuta (slika 4). Važno je primijetiti kako sličnost dobivenog modela originalnom modelu uvelike ovisi o načinu popunjavanja nastale rupe (slika 5).



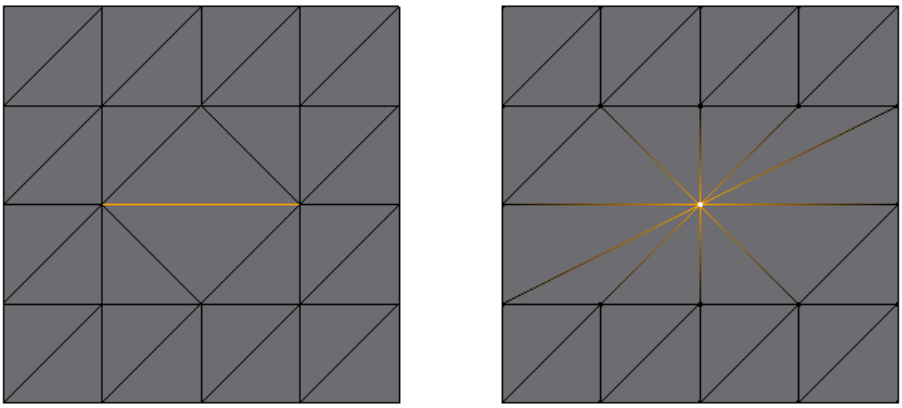
Slika 4: primjer uklanjanja vrha

2.4.2 Urušavanje ruba

U postupku urušavanja ruba (engl. *edge collapse*) [1] vrhove odabranog ruba spajamo u jedan novi vrh koja se nalazi u njegovoj sredini odnosno na jednakoj udaljenosti od krajnjih vrhova (slika 6). Preostala 2 ruba trokuta kojima je pripadao urušeni rub se spajaju u jedan rub, pa se tako sa svakim urušavanje ruba uklanjaju 2 trokuta iz modela, ali i gubi 5 rubova te dobivaju 2 nova.



Slika 5: primjer različitih načina popunjavanja rupa

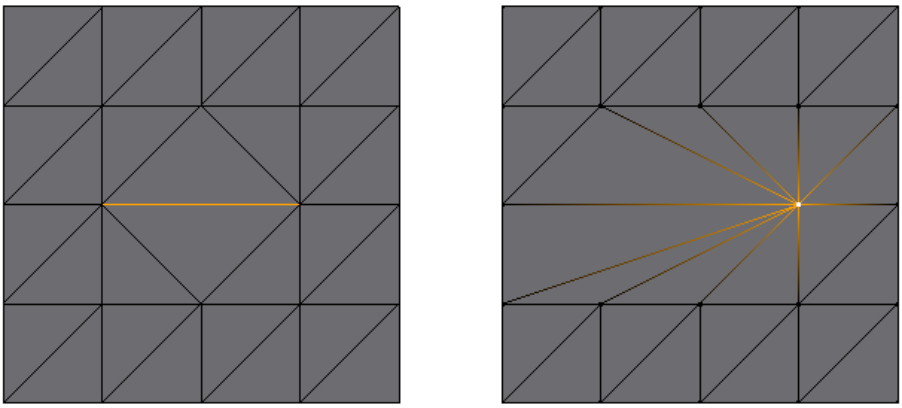


Slika 6: primjer urušavanja ruba

2.4.3 Urušavanje polu-ruba

Prilikom zapisa modela često se uz pojam ruba uvodi i njegova varijanta, polu-rub (eng. half-edge). Polu rubovi se kao i rubovi sastoje od dva povezana vrha, ali su

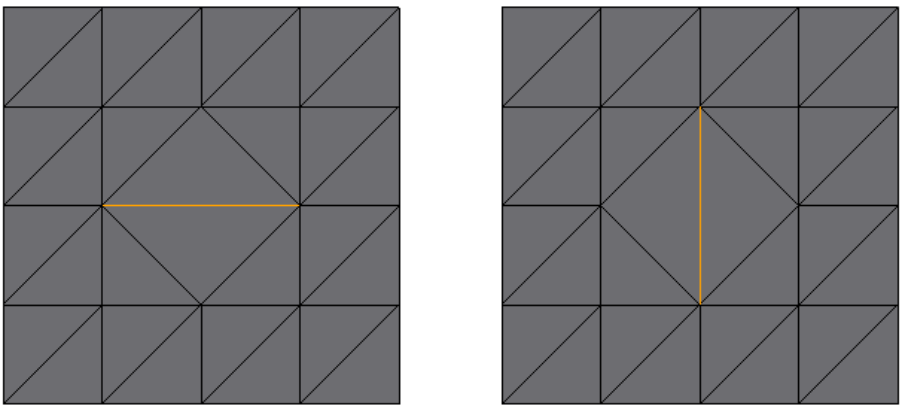
usmjereni. Ako rub povezuje točke A i B onda se on može rastaviti na dva polu-ruba od kojih jedan ima za početni vrh A a završni vrh B a drugi je suprotnog smjera odnosno početni vrh mu je B a završni A. Prilikom urušavanja polu-ruba se početna točka vrha spaja sa završnom točkom (slika 7). Prednost ovog postupka je što se ne mijenja položaj vrhova već se samo neki vrhovi odbacuju.



Slika 7: primjer urušavanja polu-ruba

2.4.4 Okretanje ruba

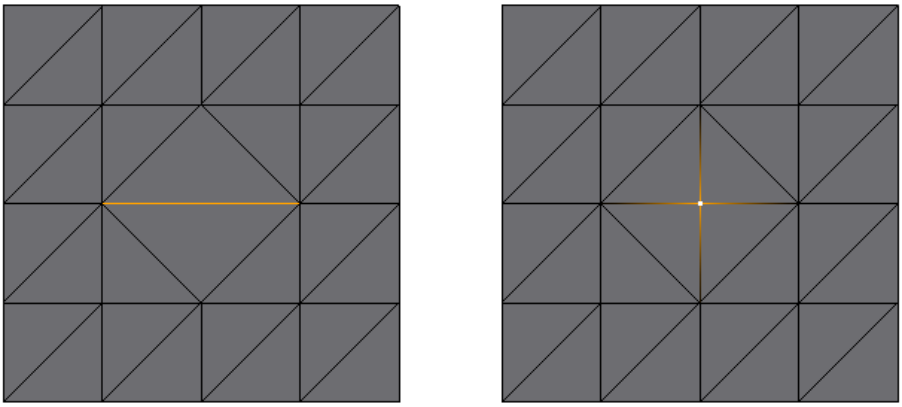
Uz uklanjanje elemenata modela postoje i nedestruktivne modifikacije modela, a jedna od njih je okretanje ruba. Ako rubu koji je okružen s 2 trokuta zamijenimo početnu točku sa sljedećom točkom u trokutu kojem ona pripada, i krajnju točku sa sljedećom točkom u trokutu kojem pripada, dobivamo novi zakrenuti rub (slika 8). Ova operacija je važna jer, slično kao i različiti načini popunjavanja rupe u modelu (slika 5), znatno može utjecati na izgled površine modela.



Slika 8: primjer okretanja ruba

2.4.5 Lomljenje ruba

U ovom postupku dodaje se novi vrh unutar postojećeg ruba te se on, pomoću dva nova ruba, povezuje s najbližim vrhovima koji već nisu spojeni s njim (slika 9) [3]. Važno je primijetiti kako ovim postupkom dodajemo dva nova trokuta u model.



Slika 9: primjer lomljenja ruba

2.5 Ocjenjivanje sličnosti modela

Kako bi odredili koliko je kvalitetno model pojednostavljen , potrebno je s njim usporediti originalni model. Kako su oba modela jednako orijentirana i generalno bi trebali biti dosta slični ovaj problem je jednostavniji od općenite usporedbe sličnosti dva različita modela različitih veličina i orijentacija.

2.5.1 Razlika površina

Vrlo jednostavan način je usporedba ukupne površine originalnog i pojednostavljenog modela. Ova metoda je iznimno brza, ali ne može garantirati sličnost modela. Ako su površine vrlo različite, pojednostavljeni model je sigurno loša aproksimacija originalnog modela. Ali i vrlo slični iznosi površina nisu dovoljan uvjet za garantiranje sličnosti modela, s toga ova metoda sama po sebi nije dovoljno dobra, no može služiti kao eliminacijska metoda prije korištenja neke kompleksnije i vremenski zahtjevnije metode.

2.5.2 Udaljenost vrhova do površine

Kao mjeru sličnosti možemo uzeti sumu udaljenosti svakog vrha originalnog modela od pojednostavljenog modela. Ova mjera je dosta precizna, ali se mogu pojaviti problemi

ako pojednostavljeni model ima deformacije na površini koja je udaljena od vrhova originalnog modela. Ipak ovakvi slučajevi su mogući samo ako originalni model ima rijetko postavljene vrhove, odnosno ima velike površine na kojima su razapeti trokuti s vrlo udaljenim vrhovima.

3 Evolucijski algoritmi

Evolucijski algoritmi su optimizacijski algoritmi inspirirani prirodom evolucijom. Osnovni element evolucijskih algoritama su jedinke, koje u pravilu predstavljaju pojedina rješenja nekog problema. Jedinke prolaze biološke mehanizme kao što su reprodukcija, mutacija i selekcija. Prilikom evolucije jedinke se ocjenjuju unaprijed definiranom funkcijom dobrote te se na temelju nje eliminiraju jedinke koje predstavljaju loša rješenja, a zadržavaju one s dobrim rješenjima. Svaka jedinka ima zapisano rješenje u obliku genotipa koji može biti proizvoljnog oblika, ali se najčešće koristi binarna vrijednost, niz cijelih ili realnih brojeva, permutacijski vektor ili stablo. Početna generacija se najčešće stvara nasumično, iako se mogu iskoristiti npr. pohlepni algoritam ili rješenja dobivena nekom drugom metodom. Sljedeće generacije se dobivaju križanjem jedinki iz prethodne generacije, odabiru se jedinke koje su kvalitetno ocijenjene funkcijom dobrote te se stvara nova jedinka koja sadrži genotip koji je mješavina genotipova njegovih roditelja.

Genetski algoritmi su podskupina evolucijskih algoritama u kojima jedinke predstavljaju rješenja neke funkcije, postupak traženja optimalnog rješenja istovjetan je onom kod evolucijskih algoritama. Za razliku od njih u *genetskom programiranju* jedinke ne predstavljaju zasebna rješenja, već računalne programe. Programi su najčešće relativno jednostavni i zapisuju se u obliku stabla.

Evolucijski algoritmi su heuristički, pa kao takvi ne garantiraju pronalazak optimalnog rješenja. U pravilu su spori ali dostižni, dobro se snalaze kod optimizacija funkcija s velikim brojem lokalnih optimuma i kada je potrebno zadovoljiti više mjera dobrote.

4 Primjena GA na optimizaciju modela

U sklopu rada ispitano je nekoliko načina primjena genetskih algoritama na optimizaciju modela. Ispitane metode se razlikuju po načinu prikaza jedinki kao i dijelu postupka uklanjanja poligona u kojem se koristi genetski algoritam.

4.1 Prva inačica

Prvi pokušaj korištenja genetskog algoritma za optimizaciju modela uključivao je prikaz jedinki kao zasebnih modela. Početna generacija se sastoji od nasumično generiranih modela, odnosno N nasumično generiranih vrhova koje su povezane na nasumičan način u trokute. Broj vrhova u generiranim modelima mora biti manji od broja vrhova u originalnom modelu kako bi postupak uistinu smanjio broj trokuta. Modeli su spremljeni u osnovnoj strukturi podataka (popis vrhova i trokuta), te se prilikom križanja nova jedinka stvara uzimanjem dijela vrhova i trokuta iz svakog od njegovih roditelja. Ova metoda se vrlo brzo pokazala izrazito neučinkovitom. Zbog vrlo velikog područja pretraživanja potrebne su vrlo velike populacije, a zbog prirode jedinki, odnosno zbog toga što sadržavaju cijele modele, jedinke su vrlo velike pa tako dolazi do ogromne potrošnje memorije. Također evaluacija je spora, što predstavlja još jedan problem s performansama. Iz navedenih razloga metoda nije u stvarnom vremenu uspjela dati rezultate koji sliče originalnom modelu.

4.2 Druga inačica

Kako se spremanje cijelog modela pokazalo neučinkovitim, pojavila se potreba za pronalaskom drugačijeg zapisa jedinki. Osnovna ideja druge generacije algoritma je spremanje popisa rubova koje je potrebno urušiti. Korištenje osnovne strukture podataka za spremanje modela nije dovoljno dobro za ovaj postupak, pa su u ovom postupku modeli spremljeni kao mreža polu-rubova. Iako se ovaj postupak pokazao znatno učinkovitijim, pojavili su se i novi problemi. Prikaz jedinki se pokazao zahtjevnim problemom: prvi pokušaj je bio korištenje liste cijelih brojeva, gdje je svaki cijeli broj predstavljao redni broj ruba kojeg je potrebno urušiti, ali urušavanje ruba uzrokuje velik broj promjena na modelu, od kojih je najproblematičniji taj što uklanja ukupno 5 rubova i stvara 2 nova. Kako se novi rubovi dodaju na kraj popisa rubova, značenje

kraja genotipa uvelike ovisi o tome koji su rubovi urušeni na početku genotipa, jer se svakim urušavanjem mijenjaju redni brojevi rubova.

Ovaj problem je naizgled lako riješiti korištenjem referenci na rubove u genotipu umjesto njihovih rednih brojeva, ali u ovom slučaju opet dolazi do problema: prvi je to što će genotip vrlo često referencirati rub koji je već uklonjen pri nekom prethodnom urušavanju, a drugi je to što genotip nikada neće referencirati neki rub koji je stvoren prilikom urušavanja nekog prethodnog ruba. Osim što je ovaj način zapisa jedinki izrazito neučinkovit, pojavljuje se i problem što je iznimno teško garantirati kako će postupak ukloniti zadani broj trokuta te ga kao takvog nije moguće koristiti.

4.3 Treća inačica

Kako se direktno referenciranje rubova pokazalo iznimno nestabilnim, u sljedećem pokušaju za definiranja jedinki se rubovi referenciraju indirektno, preko trokuta. Prilikom svakog urušavanja ruba se gube točno 2 trokuta i ne stvaraju se novi što je puno povoljnija situacija. Kao genotip se koristi niz cijelih brojeva čija je duljina jednaka broju trokuta u modelu. Svakom trokutu je dodijeljen jedan cijeli broj. Nula označava kako se taj trokut ne dira, 1, 2 i 3 označavaju redni broj stranice (ruba) trokuta se treba urušiti. Ovdje se ponovo pojavljuje problem referenciranja trokuta koji je već uklonjen, ali se u tom slučaju jednostavno taj korak preskoči budući da tog trokuta već nema pa nije potrebno ništa učiniti.

Ova inačica se pokazala znatno učinkovitijom od prethodne, ali ipak performanse su i dalje ostale problem. Za ocjenjivanje svake jedinke je potrebno kopirati cijeli originalni model, zatim ukloniti trokute iz njega i na kraju ga usporediti s originalnim modelom što je vremenski zahtjevno. Ova metoda se pokazala učinkovitom i upotrebljivom isključivo na modelima s malim brojem trokuta, dok je za veće modele vrijeme izvođenja predugo.

4.4 Četvrta inačica

4.4.1 Aproksimacija globalne pogrešle

Kako bi se smanjilo vrijeme evaluacije jedinki, umjesto točne mjere sličnosti originalnog i pojednostavljenog modela uvodi se aproksimacija pogreške nastale uklanjanjem

trokuta [2]. Za primjenu aproksimacije bilo je potrebno promijeniti pristup uklanjanja trokuta, pa se tako umjesto urušavanja rubova u trećoj inačici genetskog algoritma koristi uklanjanje vrhova. Prilikom uklanjanja vrhova uklanjaju se i svi trokuti vezani za njega, a zatim se nastala rupa popunjava.

Kako bi izračunali aproksimaciju globalne pogreške potrebno je prvo definirati aproksimaciju lokalne pogreške koja se računa za pojedini vrh [2]. Svakom vrhu možemo pridružiti normalu koju računamo tako da zbrojimo normale trokuta kojima taj vrh pripada i zatim normaliziramo dobiveni vektor. Prilikom računanja lokalne pogreške u nekom vrhu prvo stvorimo ravninu koja ima normalu jednaku normali tog vrha i ujedno prolazi tim vrhom. Zatim računamo udaljenost svake točke koja je susjedna vrhu čiju pogrešku računamo od ravnine tog vrha. Na ovaj način računamo koliko je vrh izbočen, odnosno kolika je zakrivljenost u njegovoj okolini, što nam daje dobru procjenu koliko će uklanjanje tog vrha utjecati na površinu njegove okoline. Za računanje aproksimacije globalne pogreške potrebno je spremati aproksimaciju globalnih pogrešaka za pojedine trokute modela. Na početku su sve globalne pogreške trokuta postavljene na 0, a algoritam prolazi kroz vrhove koji se planiraju ukloniti [2]. Za svaki vrh se računa aproksimacija lokalne pogreške, zatim se u trokutima kojima taj vrh pripada traži onaj s najvećom globalnom pogreškom, dobivena globalna pogreška se zbraja s aproksimacijom lokalne pogreške trenutnog vrha te se dobivena vrijednost postavlja kao nova aproksimacija globalne pogreške za sve trokute kojima trenutni vrh pripada. Na kraju postupka aproksimacija globalne pogreške je jednaka najvećoj globalnoj pogrešci svih trokuta u modelu.

4.4.2 Jedinke i križanje

U ovom postupku je vrlo povoljno i to što uklanjanje jednog vrha nikako ne utječe na druge vrhove, pa je zbog toga lako definirati genotip. Genotip se sastoji od niza Booleovih varijabli, a duljina niza je jednaka broju vrhova u modelu, tako da je svakom vrhu pridružena jedna vrijednost (“istina” ili “laž”) koja označava je li taj vrh potrebno ukloniti. Prilikom stvaranja početne generacije sve jedinke će imati N/2 istinitih vrijednosti u svom genotipu, gdje je N jednak broju trokuta koje treba ukloniti. Prilikom križanja se koristi jednostavno križanje s jednom prekidnom točkom, nakon čega se provjerava broj istinitih vrijednosti u genotipu. Ako je on veći od N/2 nasu-

mična istinita vrijednost iz genotipa se pretvara u laž, s druge strane ako je on manji nasumična lažna vrijednost se pretvara u istinu. Ovaj postupak se ponavlja sve dok je broj istinitih vrijednosti različit od N/2.

4.4.3 Popunjavanje rupa

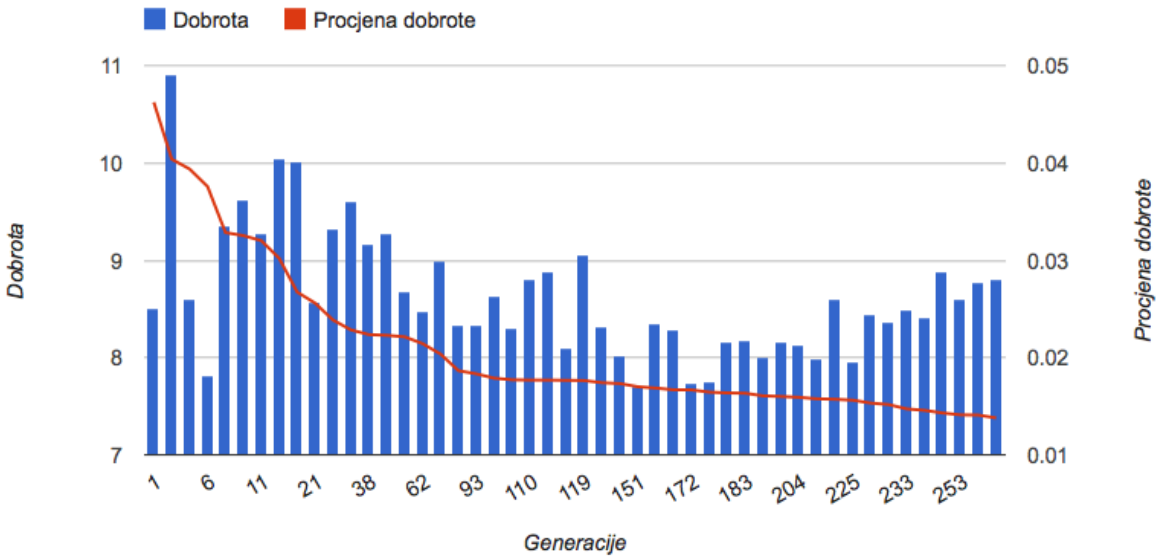
U prvoj varijanti algoritma se prilikom popunjavanja rupe prihvaćalo prvo valjano rješenje koje je nađeno, što je dovodilo do nekvalitetnih rezultata. Implementacijom pohlepnog algoritma koji traži povoljno rješenje rezultati su se znatno poboljšali te se pokazalo kako metoda popunjavanja ima značajan utjecaj na kvalitetu pojednostavljenog modela. Za rad pohlepnog algoritma potrebno je prvo pronaći jedno valjano rješenje, a zatim se slijedom okretanja rubova pokušava naći povoljnije rješenje. Algoritam slijedno okreće svaki rub, te ako je njegovo okretanje dovelo do poboljšanja ostavi ga okrenutog, a ako nije vraća ga nazad u početnu poziciju. Postupak se prekida kad niti jedno okretanje u iteraciji ne poboljša kvalitetu novih trokuta.

4.4.4 Rezultati

Nažalost, metoda aproksimacije globalne pogreške pokazala se nedovoljno preciznom za korištenje u genetskom algoritmu. Iako prilikom evolucije aproksimacija pogreške konstantno pada, prava pogreška ne pokazuje slično ponašanje (slika 10) već varira gotovo neovisno.

Iako su dobiveni modeli slični originalnima njihova kvaliteta nije ni približno zadovoljavajuća, a vrijeme izvođenja za kompleksnije modele, iako je znatno manje nego u prethodnoj inačici, i dalje nije prihvatljivo. U tablici 4.1. su iznesena prosječna vremena izvođenja za pojednostavljivanje modela ptice sa 1394 trokuta na 400 trokuta, izražena u sekundama, za različite veličine početne populacije. Dobrota modela je izračunata metodom opisanom u poglavlju 2.5.2. Mjerenja su izvršena 30 puta. Važno je primjetiti da već pri relativno malim populacijama vrijeme izvođenja raste na preko 3 minute što nije prihvatljivo s obzirom na kvalitetu dobivenih modela.

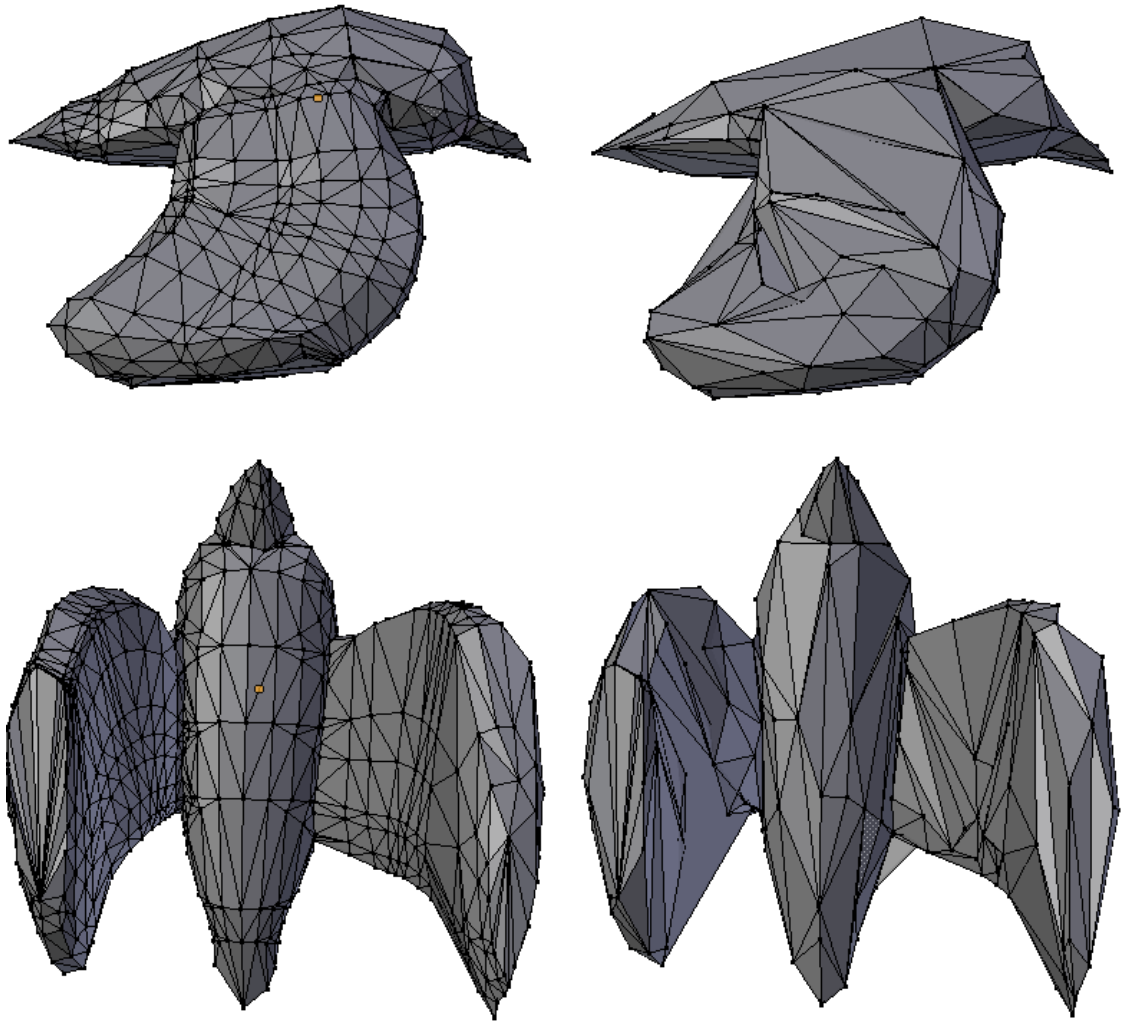
Na slici 11 je vidljiv rezultat izvođenja za četvrto mjerenje. Osnovne karakteristike modela su sačuvane ali model je daleko od optimalnog. Također vidljiv je i problem s popunjavanjem nastalih rupa, jer se novi trokuti preklapaju i prolaze jedan kroz drugog. Ovaj problem se pokušava riješiti u petoj inačici programa.



Slika 10: usporedba dobrote i procjene dobrote

Mjerenje	Početna populacija	Broj djece	Vrijeme izvođenja	Dobrota
1	10	5	21.87	8.858
2	25	10	39.12	9.056
3	100	50	91.86	8.356
4	500	250	185.63	8.252

Tablica 4.1: rezultati mjerenja za četvrtu inačicu GA



Slika 11: originalni model (lijeva strana) i primjer rezultata četvrte inačice GA (desna strana)

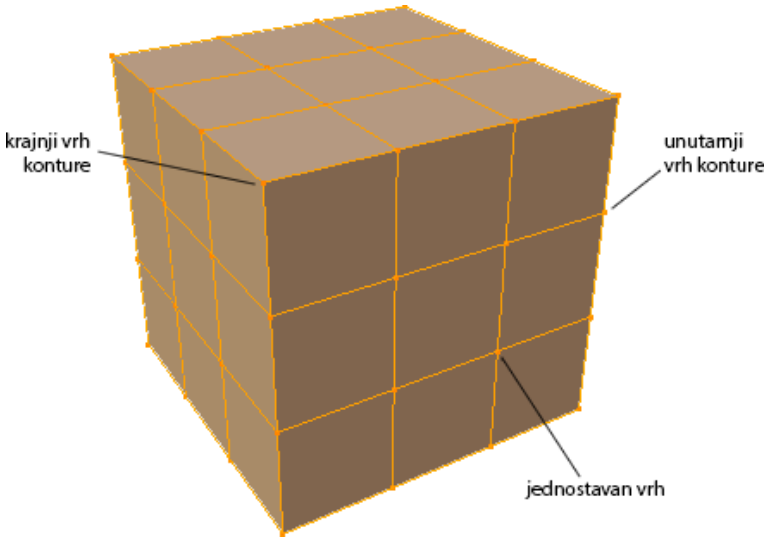
4.5 Peta inačica

Pronalazak optimalnog načina popunjavanja rupa nastalih brisanjem vrhova se pokazao kao kompleksan problem za kojeg pohlepni algoritam nije dovoljno dobar. Osim što je potrebno osigurati sličnost novih trokuta s trokutima koji su postojali prije brisanja vrha, poželjno je zadovoljiti i druge uvijete, kao što su izbjegavanje preklapanja trokuta i održavanje umjerene kardinalnosti vrha. Kardinalnost vrha se definira kao broj rubova koji su s njime povezani. Nije poželjno imati veliku kardinalnost vrha jer se njegovim brisanjem uklanja veliki broj trokuta, pa tako nastaje veća rupa koju je teže kvalitetno popuniti. S druge strane, kako bi mogli prikazati model, svaki vrh mora imati kardinalnost minimalno 3. Zbog toga nije poželjno pretjerano smanjiti kardinalnosti vrhova jer se time ograničava broj dozvoljenih akcija. Također okretanje jednog ruba utječe na ostale rubove, pa je tako bitan i redosljed njihovog okretanja. Zbog toga se pojavila ideja korištenja genetskog algoritma za pronalaženje optimalnog načina popunjavanja rupa nastalih brisanjem vrhova.

4.5.1 Odabir vrhova za uklanjanje

Kako bi održali prihvatljive performanse, u ovoj inačici algoritma se za izbor vrhova koje je potrebno ukloniti koristi pohlepni algoritam zasnovan na aproksimaciji pogreške, a genetski algoritam se koristi samo za popunjavanje nastalih rupa. Prilikom izbora vrha za uklanjanje najprije se određuje vrsta svakog vrha [6]. Vrh može biti jednostavan, unutarnji vrh konture ili krajnji vrh konture (slika 12). Kako bi definirali svaku od navedenih vrsta vrha, prvo trebamo odrediti granični kut, koji može biti proizvoljne veličine, a unutar rada je izabran kut od 120 stupnjeva. Ako su trokuti nekog ruba pod kutem manjim od graničnog kuta taj rub ćemo nazvati *oštrim* rubom. Vrhove klasificiramo s obzirom na broj oštrih rubova koji iz njega izlaze. Ako nema oštrih rubova taj vrh je *jednostavan* i pripada manje bitnom djelu modela. Ako vrh ima točno dva oštra ruba onda je on *unutarnji vrh* konture. Rubovi koji imaju jedan ili više od dva oštra ruba su *krajnji vrhovi* konture, odnosno to su rubne točke modela.

Krajnji vrhovi konture se nikada ne uklanjaju i njih se u procesu odabira jednostavno odbacuje, dok su ostali vrhovi kandidati za uklanjanje. U svakom koraku se određuje dobrota svakog vrha i uklanja onaj koji je najpogodniji. Dobrota jednostavnih vrhova se određuje kao prosjek kvadrata udaljenosti susjednih vrhova od ravnine



Slika 12: vrste vrhova

vrha kojeg uklanjamo. Prosjek se računa kako bi dobrota bila usporediva drugim vrhovima, bez obzira koliko drugi vrhovi imaju susjeda. Ako je odabran jednostavan vrh za uklanjanje on se briše i daljnjim postupkom se nastala rupa popunjava.

Prilikom uklanjanja unutarnjeg vrha konture potrebno je zadržati rub konture. Prvo se određuju dva susjedna vrha koji su dijelovi te konture, zatim se briše središnji vrh te stvara novi rub koji povezuje preostala dva vrha konture. Ovim postupkom dobivamo dvije manje rupe, odnosno moramo riješiti dva manja problema popunjavanja rupa. Zbog ovog postupka određivanje dobrote vrhova konture je nešto drugačije, kod njih računamo kvadrat udaljenosti vrha kojeg uklanjamo od dužine koja spaja dva susjedna vrha konture.

4.5.2 Funkcija dobrote

Prije uklanjanja vrha spremamo njegovu poziciju kao i poziciju centra svih trokuta koji pripadaju tom vrhu. Te vrijednosti se koriste za ocjenu sličnosti originalnih trokuta s obzirom na nove trokute kojima smo popunili rupu. Za svaku točku se računa udaljenost od novih trokuta, a kao ocjena sličnosti se uzima prosjek kvadrata tih udaljenosti. Uz to se odbacuju sva rješenja koja stvaraju novu konturu, odnosno sadrže rub čiji trokuti zatvaraju kut manji od zadanog graničnog kuta, jer se čuvanje postojećih kontura osigurava u prethodnom koraku pa nema potrebe za stvaranjem novih kontura.

4.5.3 Jedinke i križanje

Genotip svake jedinke sadrži popis rubova koje je potrebno okrenuti. Jedan rub se u genotipu može pojaviti više puta zato što okretanje rubova može promijeniti okolinu rubova koji su već prethodno okrenuti. Prilikom križanja uzima se nasumično veliki dio s početka prvog roditelja i nasumično veliki dio sa završetka drugog roditelja te se njihovim spajanjem stvara nova jedinka.

4.5.4 Populacija i evolucija

Veličina populacije ovisi o broju rubova koji su nastali popunjavanjem rupe. Prilikom evolucije nove jedinke se dodaju u trenutnu populaciju te se zatim zadržava zadani broj najboljih jedinki a ostale se odbacuju. Roditelji se biraju turnirskom metodom, odnosno biraju se nasumično tri jedinke, najgora se odbacuje a preostale dvije postaju roditelji.

4.5.5 Podešavanje parametara

Kako bi osigurali što je moguće bolje rezultate potrebno je podesiti parametre evolucije. Cilj je dobiti modele što sličnije originalnom modelu, ali je također potrebno zadovoljiti i vremenska ograničenja budući da su postojeći algoritmi u pravilu iznimno brzi. Mjerenja se izvode na modelu ptice s 1394 trokuta koji se smanjuje na 400 trokuta, a ciljano vrijeme izvođenja je ispod 60 sekundi.

Veličina populacije i broj novih jedinki po generaciji su prvi parametri koji su optimirani. Kako težina problema popunjavanja ovisi o veličini rupe koju je potrebno popuniti, ove parametre je moguće definirati kao izraz koji ovisi o N, gdje je N broj rubova u rupi koja se trenutno popunjava. Za mjerenje je iskorišten model ptice s 1394 trokuta, kojeg je potrebno pretvoriti u model s 400 trokuta. Za svaku kombinaciju parametara mjerenja su ponovljena 30 puta, a rezultati se sastoje od prosječne dobrote pojednostavljenog modela najboljih rješenja iz svih ponavljanja (izmjereno metodom udaljenosti vrhova originalnog modela od površine pojednostavljenog modela) i prosječnog vremena izvođenja pojednostavljivanja izraženog u sekundama. Broj generacija je ograničen na 100 za sva mjerenja.

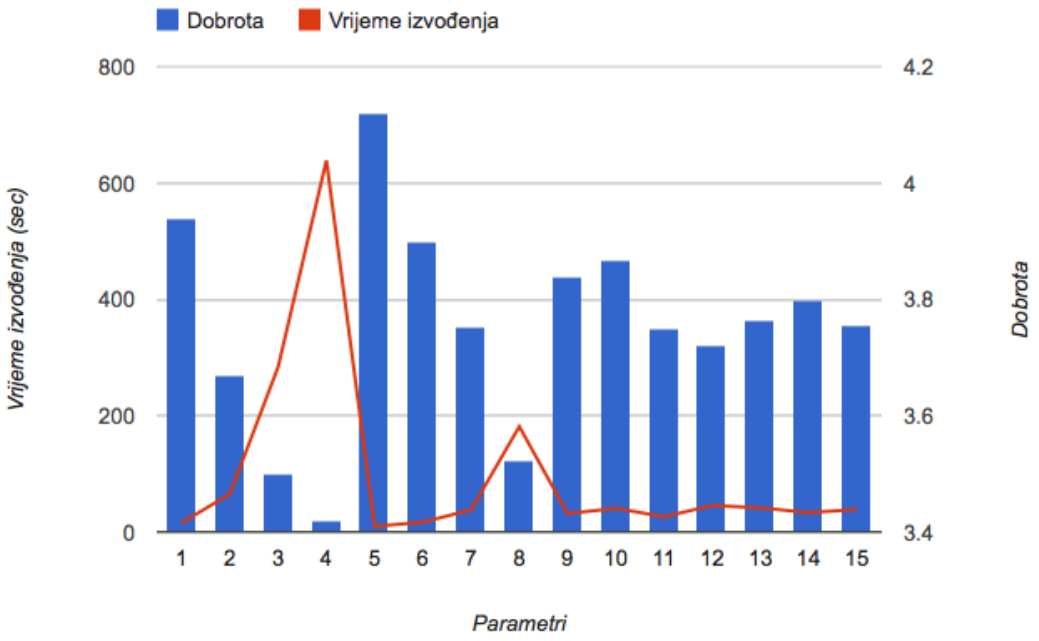
id	Početna populacija	Novih jedinki po generaciji	Dobrota	Vrijeme izvođenja
1	5	5	3.939	16.31
2	25	25	3.669	65.89
3	100	100	3.499	284.16
4	250	250	3.402	638.53
5	N	N	4.120	10.22
6	$2N$	$2N$	3.899	17.38
7	$5N$	$5N$	3.753	38.52
8	$25N$	$25N$	**3.523**	**181.61**
9	$2N + 5$	$2N + 5$	3.838	32.03
10	25	N	3.866	41.28
11	$5N$	5	3.749	26.67
12	$10N$	5	3.720	46.04
13	$5N + 10$	5	3.763	42.04
14	$2N^2$	5	3.799	33.79
15	$2N^2$	10	3.755	39.24

Tablica 4.2: rezultati mjerenja za petu inačicu GA

Rezultati su vidljivi u tablici 4.2. i grafu na slici 13.4. Kao zadovoljavajući parametri uzeti su veličina početne populacije i broj novih jedinki po generaciji $25N$ (red 8 u tablici), jer daju vrlo dobre rezultate u relativno kratkom vremenu. Ipak kako bi vrijeme izvođenja bilo usporedivo s postojećim metodama daljnjim podešavanjima potrebno je dodatno smanjiti vrijeme izvođenja na red veličine 60 sekundi.

Uvjet zaustavljanja se postavio kao drugi parametar za optimiranje. Kako vrijeme izvođenja od pribiližno 3 minute za model ove veličine nije prihvatljivo, primarni cilj optimiranja uvijeta zaustavljanja je bilo smanjivanje vremena izvođenja. Uvjet zaustavljanja ponovo može ovisti o veličini problema koji se rješava odnosno rupe koja se popunjava. Varijabla N predstavlja broj rubova u rupi koja se trenutno popunjava. Rezultati mjerenja su vidljivi u tablici 4.3. i na slici 14. Za uvijet zaustavljanja izabrano je $N + 10$ generacija bez poboljšanja (red broj 10) jer pruža najbolje rezultate u zadovoljavajućem vremenu.

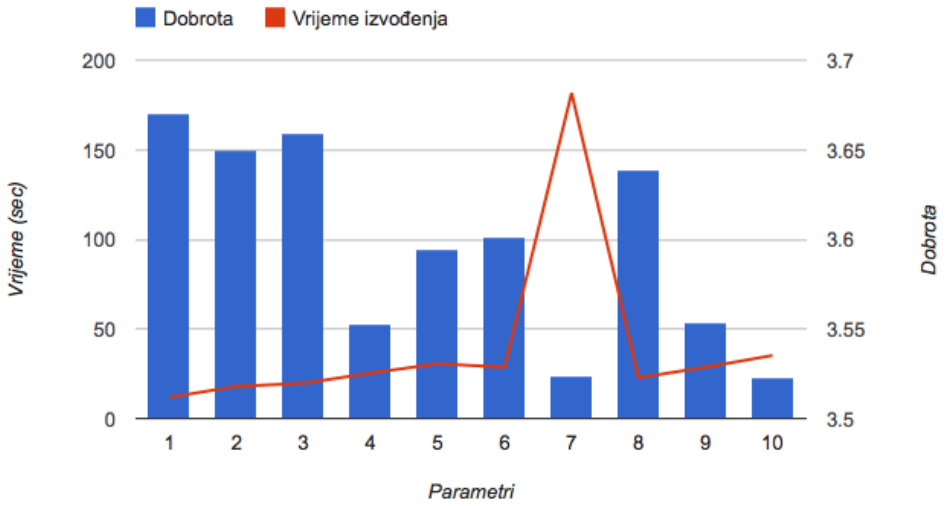
Broj gena početne populacije se nasumično generira s obzirom na veličinu problema kojeg je potrebno rješiti. Svaki gen predstavlja jedno okretanje ruba, a isti gen se može više puta pojaviti unutar genotipa, tj. isti rub se može više puta okrenuti. Rezultati su vidljivi u tablici 4.4 i na slici 15. Pokazalo se kako ovaj parametar ima značajan utjecaj na kvalitetu rezultata. Formula $rand(5N) + 5$ je odabrana jer pruža



Slika 13: rezultati mjerenja za petu inačicu GA

Id	Broj generacija	Broj generacija bez poboljšanja	Dobrota	Vrijeme izvođenja
1	5	-	3.6706	11.97
2	-	5	3.6498	18.06
3	10	-	3.6587	19.75
4	-	$5 + N$	3.5527	25.44
5	-	10	3.5943	30.64
6	15	-	3.6010	28.78
7	100	-	3.5235	181.61
8	-	$2N$	3.6390	22.84
9	-	$2N + 3$	3.5537	28.72
10	-	$N + 10$	**3.5230**	**35.27**

Tablica 4.3: rezultati ispitivanja uvjeta zaustavljanja



Slika 14: rezultati ispitivanja uvjeta zaustavljanja

id	Broj gena	Dobrota	Vrijeme izvođenja
1	$rand(N)+1$	3.523	35.27
2	$rand(2N)+1$	3.506	36.95
3	$rand(N)+5$	3.438	36.55
4	$rand(2N)+5$	3.414	36.56
5	$rand(3N)+5$	3.438	36.85
6	$rand(5N)+5$	**3.378**	**43.89**
7	$rand(5N)+10$	3.389	43.69
8	$rand(10N)+10$	3.418	49.01

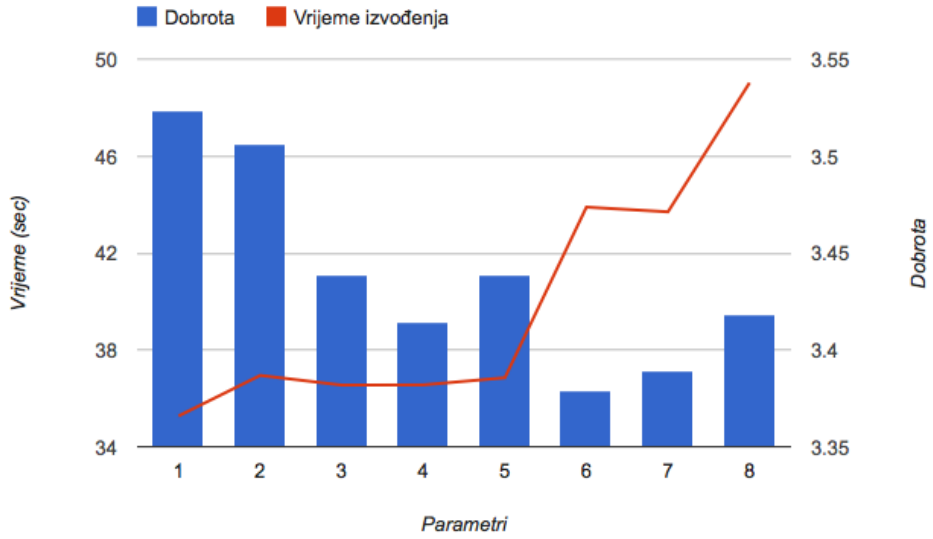
Tablica 4.4: rezultati ispitivanja broja gena početne populacije

značajno bolje rezultate uz minimalno produljenje vremena izvođenja.

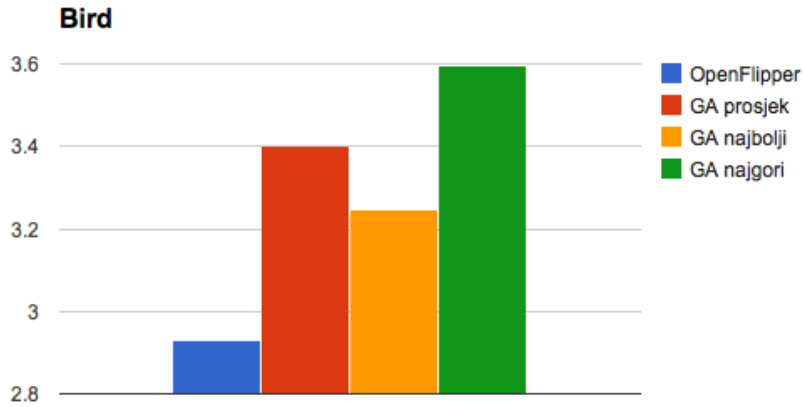
Usporedbom rezultata s rezultatima dobivenim pomoću programa OpenFlipper dobivamo uvid u realnu kvalitetu rezultata. Mjerenja su ponovljena 30 puta te su u tablici 4.5. izraženi prosječni, najbolji i najgori rezultati. Iako su rezultati dobiveni razvijenom metodom usporedive kvalitete s rezultatima dobivenim pomoću programa OpenFlipper i dalje ga ne dostižu u kvaliteti, a vrijeme izvođenja je i znatno dulje.

Model	OpenFlipper	GA prosjek	GA najbolji	GA najgori	Graf	Rezultat
bird	2.929	3.401	3.246	3.596	Slika 16.	Slika 18.
teddy	29.604	34.406	32.471	36.500	Slika 17.	Slika 19.

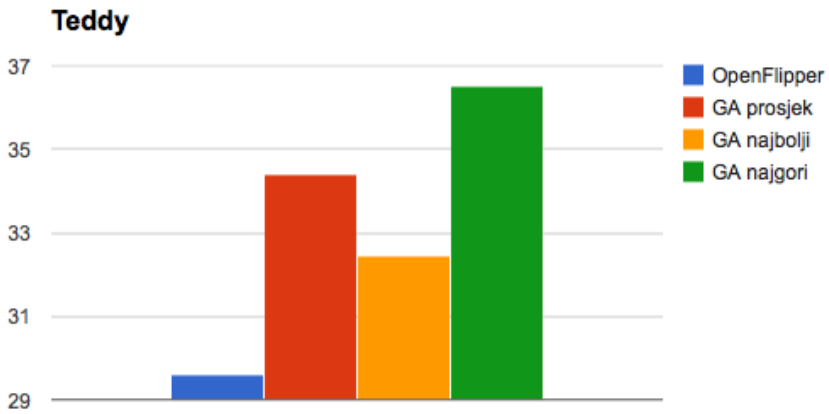
Tablica 4.5: usporedba rezultata



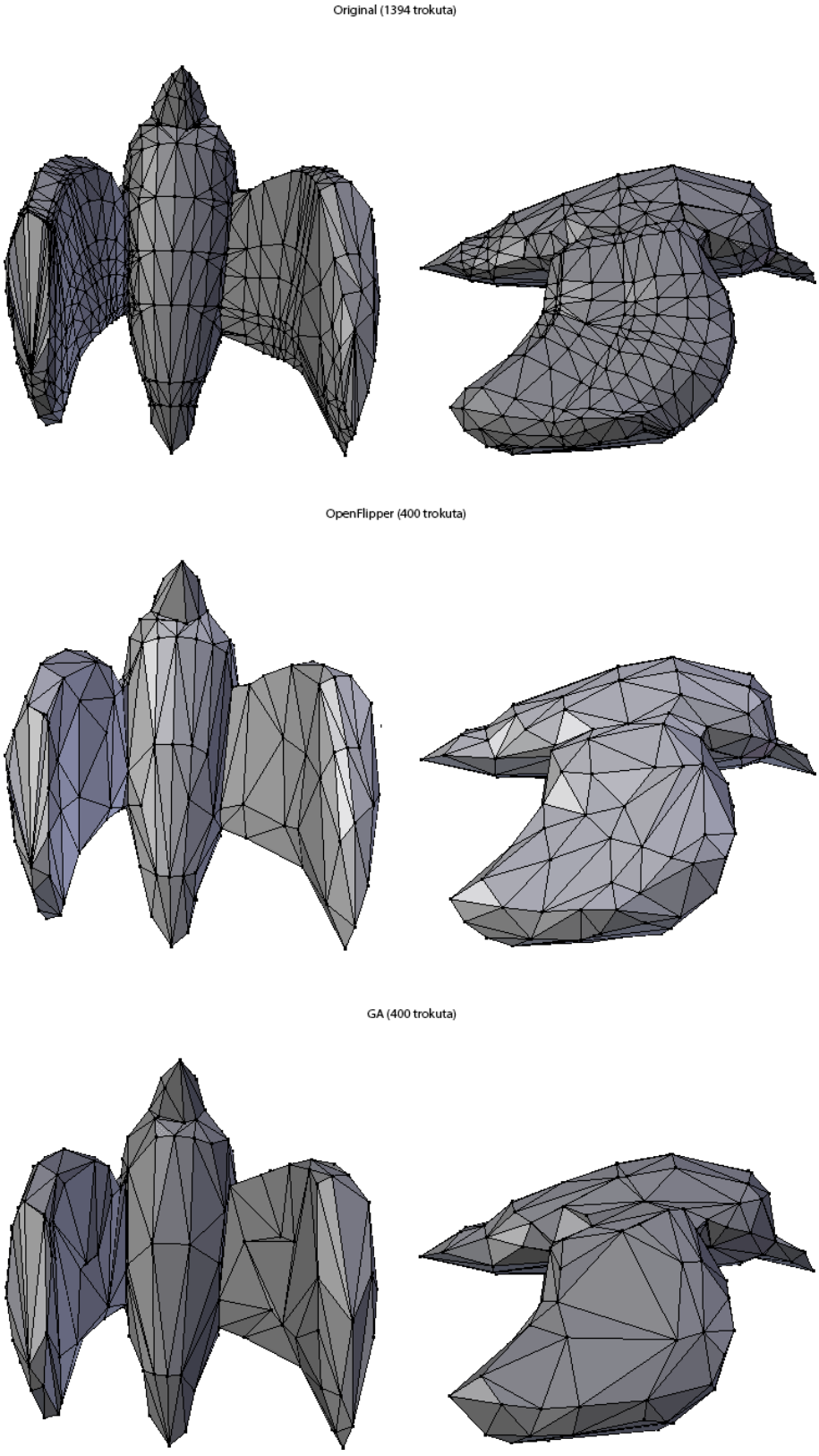
Slika 15: rezultati ispitivanja broja gena početne populacije



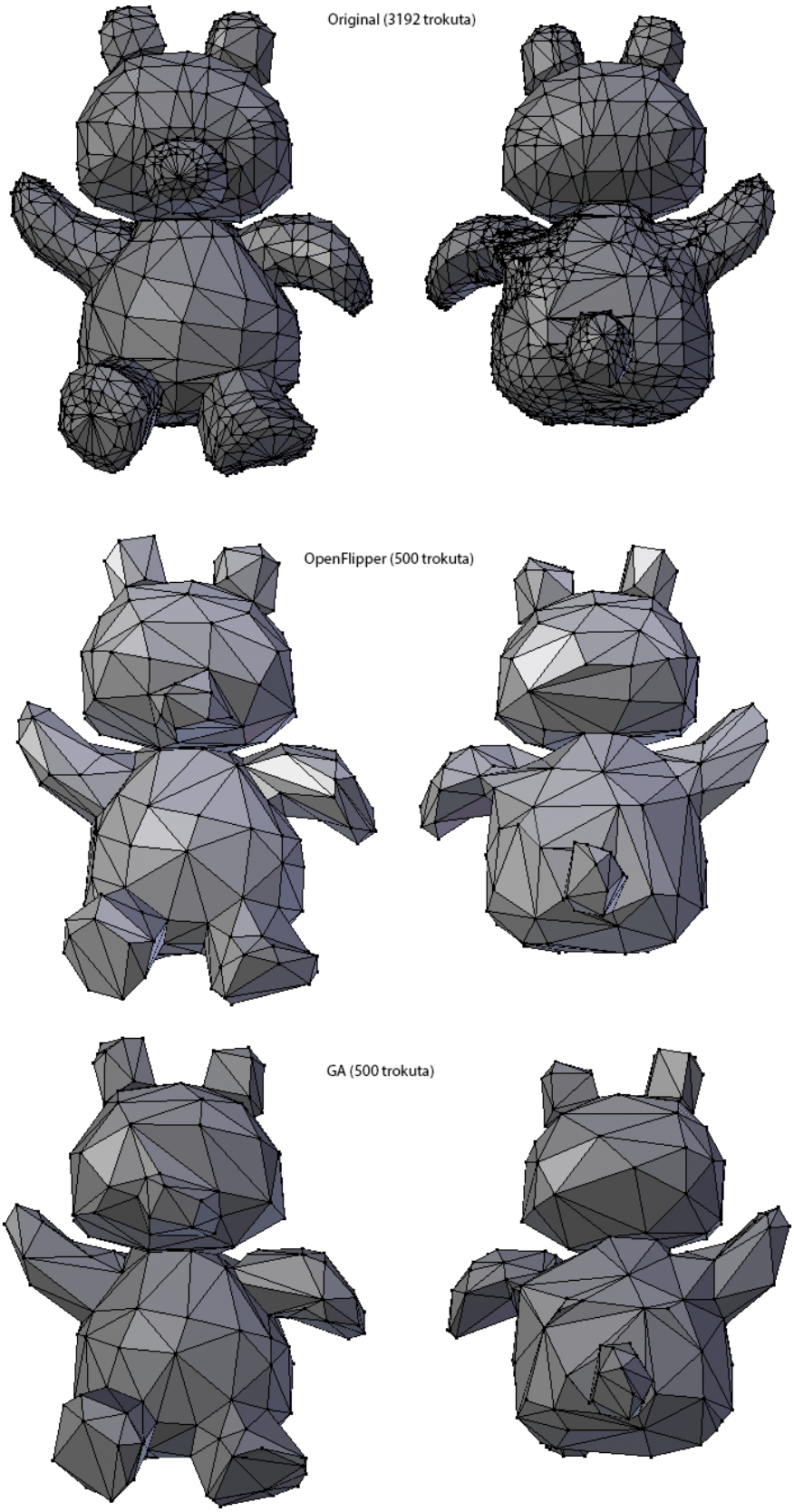
Slika 16: usporedba dobrote pojednostavljenih modela na primjeru ptice (Bird)



Slika 17: usporedba dobrote pojednostavljenih modela na primjeru medvjeda (Teddy)



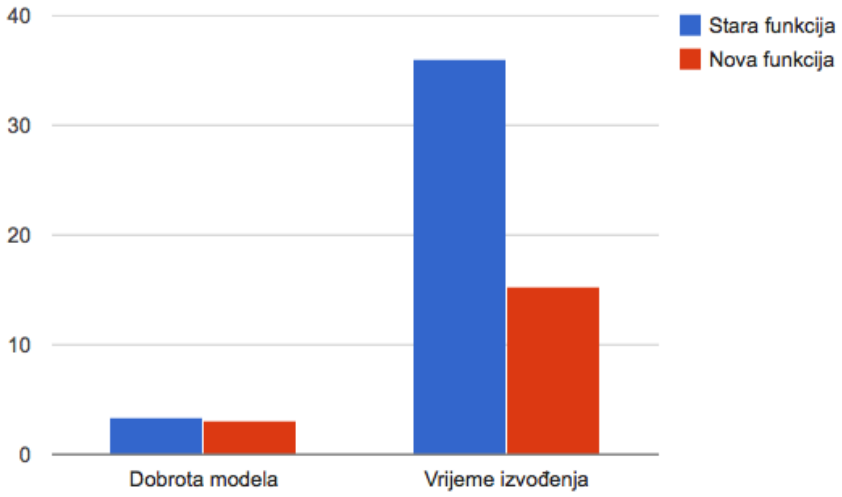
Slika 18: usporedba originalnog modela ptice s modelima dobivenim pomoću programa OpenFlipper i razvijenim GA



Slika 19: usporedba originalnog modela medvjeda s modelima dobivenim pomoću programa OpenFlipper i razvijenim GA

	Prosjek dobrote	Najbolja dobrota	Najgora dobrota	Vrijeme izvođenja
Stara funkcija	3.411	3.275	3.606	36.06
Nova funkcija	3.101	2.862	3.379	15.33

Tablica 4.6: rezultati testiranja različitih funkcija dobrote



Slika 20: rezultati testiranja različitih funkcija dobrote

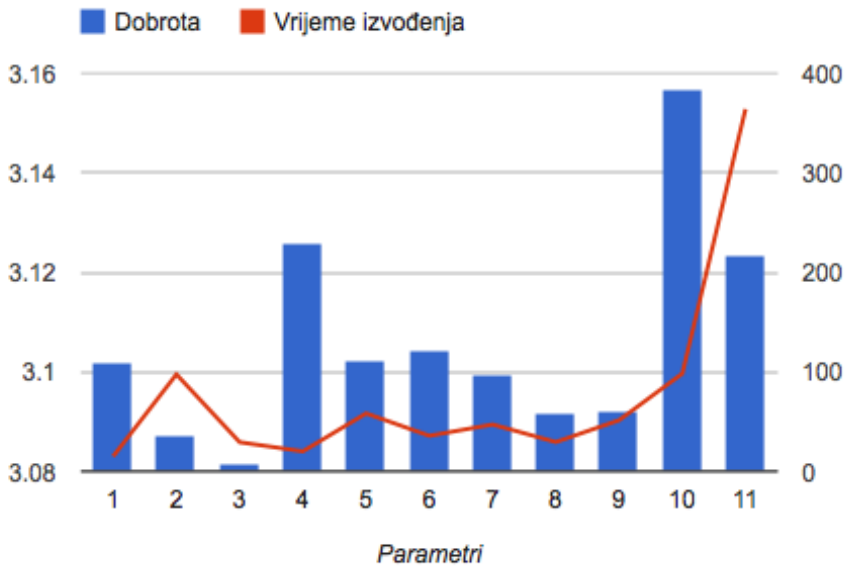
Funkcija dobrote ima važnu ulogu u optimizaciji modela pa je tako uz funkciju opisanu u poglavlju 4.5.2. isprobana još jedna varijanta. Ako za određivanje kvalitete novih trokuta koristimo njihovu udaljenost samo od uklonjene točke, a zanemarimo udaljenosti centara starih trokuta, potrebno je znatno manje računanja, a dobiveni rezultati su značajno bolji. Važno je primijetiti kako su bolji rezultati vezani uz definiciju funkcije dobrote kojom se ocjenjuje cijeli model. Naime na globalnoj razini se računa suma udaljenosti vrhova starog modela od površine novog modela, što je analogno novoj, pojednostavljenoj, lokalnoj funkciji dobrote. Mjerenja su ponovljena 30 puta a rezultati su vidljivi u tablici 4.6 i na slici 20.

Kako je vrijeme izvođenja značajno palo, postavlja se pitanje je li moguće dodatno poboljšati rezultate, povećavanjem populacije ili promjenom uvjeta zaustavljanja, uz zadržavanje prihvatljivog vremena izvođenja. Ponovljena su mjerenja s različitim veličinama populacije i uvjetima zaustavljanja. U svim primjerima je broj novih jedinki po generaciji jednak veličini početne generacije a uvjet zaustavljanja je broj generacija bez poboljšanja. Rezultati su vidljivi u tablici 4.7 i na slici 21. Kao najbolje rješenje uzeti su parametri pod brojem 3.

Ako se novi rezultati usporede s rezultatima dobivenim pomoću programa Open-

id	Populacija	Uvjet zaustavljanja	Dobrota	Vrijeme izvođenja
1	$25N$	$10 + N$	3.1018	15.33
2	$25N$	100	3.0872	97.66
3	$50N$	$10 + N$	**3.0816**	**29.58**
4	$25(N + 2)$	$10 + N$	3.1257	20.38
5	$100N$	$10 + N$	3.1024	58.48
6	$50N$	$10 + 2N$	3.1043	36.00
7	$50N$	$20 + N$	3.0995	47.33
8	150	$10 + N$	3.0919	29.70
9	150	$20 + N$	3.0921	51.63
10	500	$10 + N$	3.1567	98.43
11	500	$10 + N$	3.1233	363.86

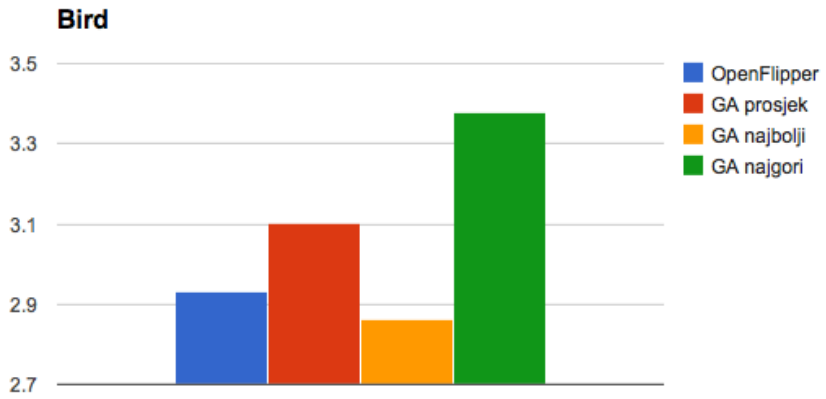
Tablica 4.7: rezultati novog testiranja veličine populacije i uvjeta zaustavljanja



Slika 21: rezultati novog testiranja veličine populacije i uvjeta zaustavljanja

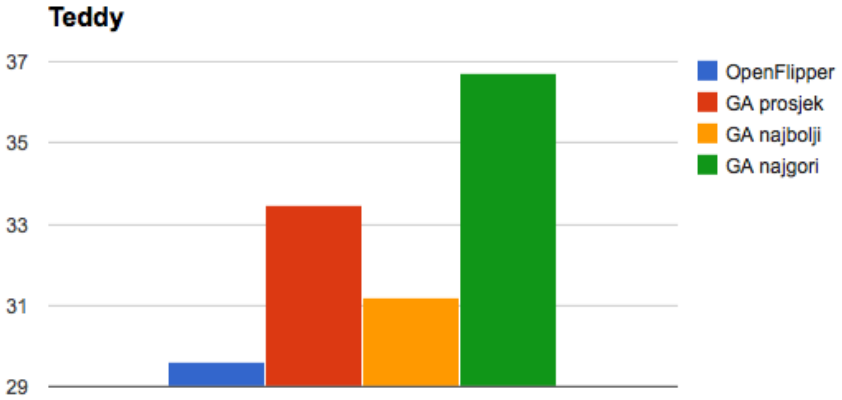
Model	OpenFlipper	GA prosjek	GA najbolji	GA najgori	Graf	Rezultat
bird	2.929	3.101	2.862	3.379	Slika 22	Slika 24
teddy	29.604	33.483	31.175	36.713	Slika 23	Slika 25

Tablica 4.8: usporedba dobrote pojednostavljenih modela

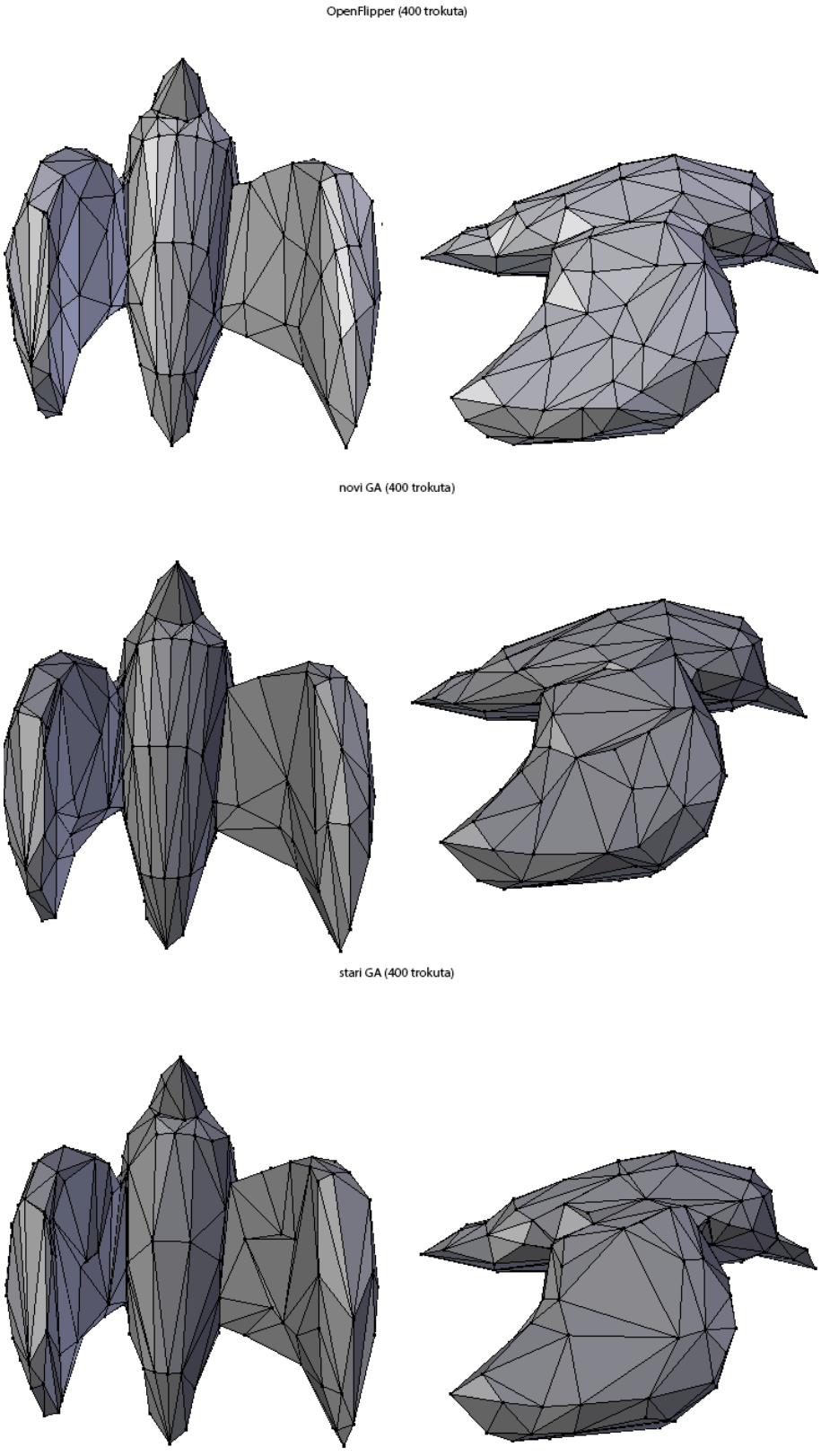


Slika 22: usporedba dobrote pojednostavljenih modela na modelu ptice (Bird)

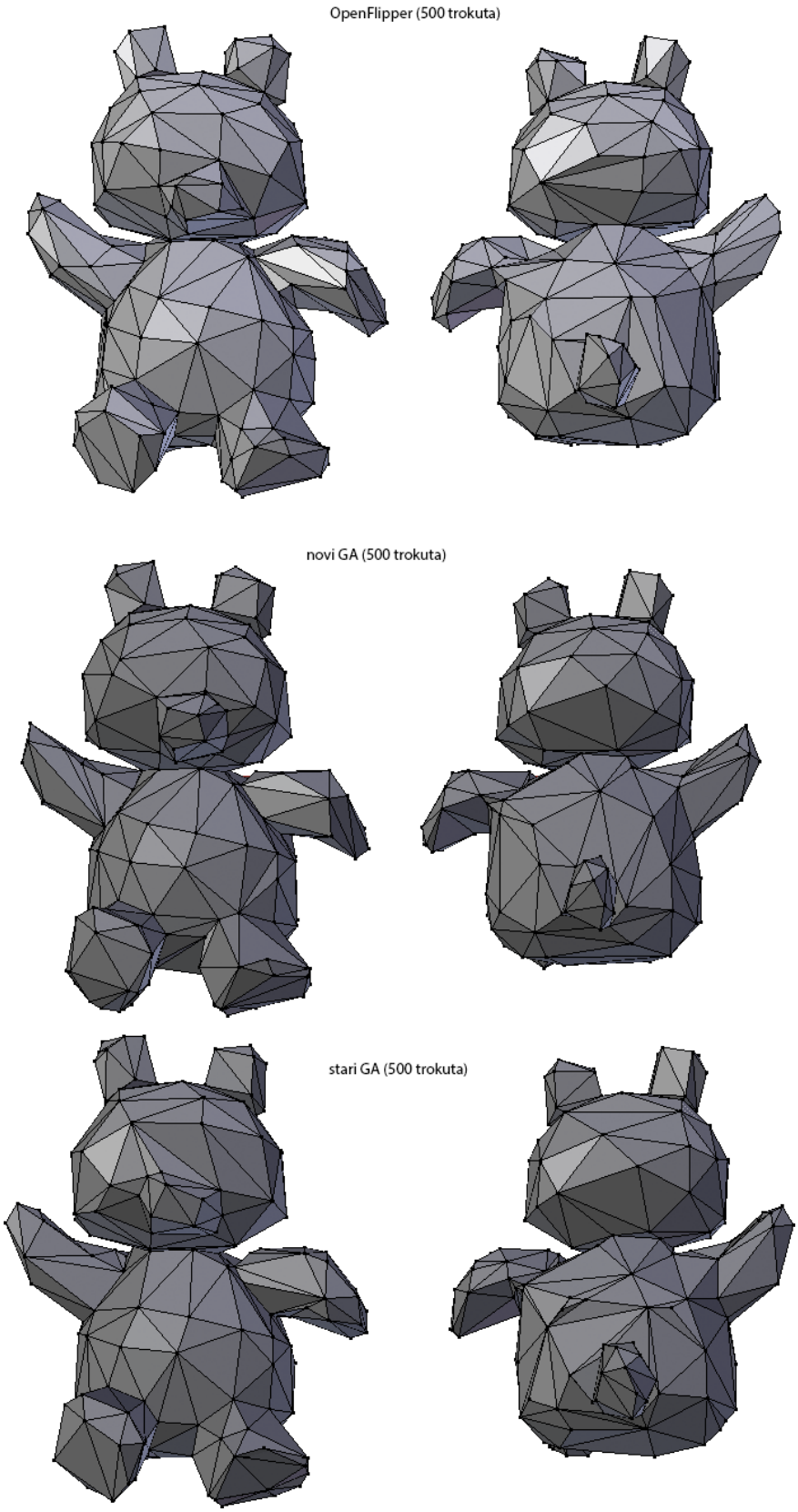
Flipper (tablica 4.8) vidljivo je kako su za model ptice dobiveni bolji rezultati, unatoč tome što je prosjek i dalje gori. Kod modela medvjeda je napredak znatno manji i kod njega OpenFlipper i dalje daje znatno bolje rezultate.



Slika 23: usporedba dobrote pojednostavljenih modela na modelu medvjeda (Teddy)



Slika 24: usporedba originalnog modela ptice s modelima dobivenim pomoću programa OpenFlipper i razvijenim GA



Slika 25: usporedba originalnog modela medvjeda s modelima dobivenim pomoću programa OpenFlipper i razvijenim GA

5 Primjena GP na optimizaciju modela

Prilikom pojednostavljivanja modela potrebno je procijeniti uklanjanje kojeg elementa će uzrokovati najmanju deformaciju modela. Kako je ovaj problem kompleksan, u sklopu rada je pomoću genetskog programiranja napravljen pokušaj razvoja funkcije za procjenu pogodnosti urušavanja ruba. Za spremanje modela se koristi mreža polurubova što omogućava jednostavno referenciranje rubova koji su susjedni rubu kojeg ocjenjujemo.

5.1 Jedinke

Genotip jedinki je zapisan u obliku stabla (slika 26) koje predstavlja funkciju za izračun pogodnosti urušavanja ruba. Generirana funkcija prima jedan argument, a to je referenca na rub čija se pogodnost ocjenjuje. Postoje 3 osnovna tipa čvorova stabla: operatori, varijable i konstante. Stabla u teoriji mogu biti beskonačno visoka, ali se zbog prirode algoritma postavlja ograničenje na njihovu visinu prilikom stvaranja početne populacije kao i prilikom križanja. Prilikom stvaranja jedinke stablo se nasumično generira, početni čvor je uvijek operator a zatim se daljnji čvorovi biraju nasumično. Ako je visina generiranog stabla veća od zadanog ograničenja, ono se odbacuje.

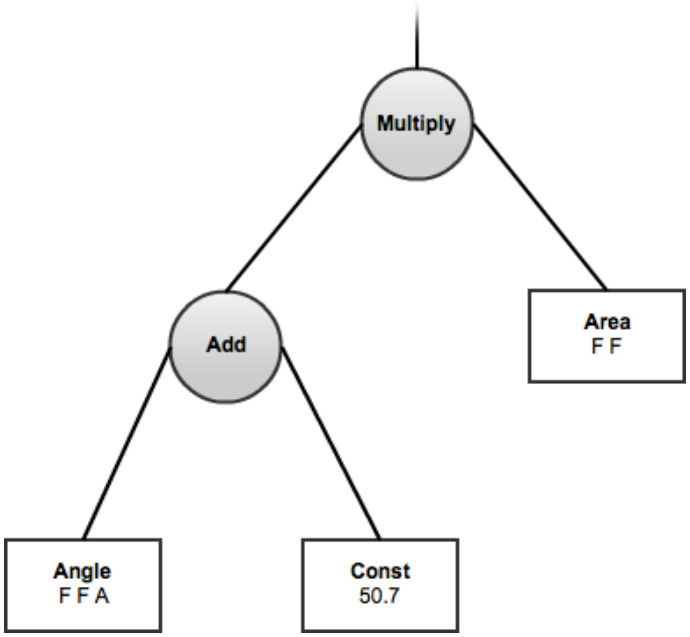
Kako bi se kvalitetne jedinke trajno sačuvale omogućeno je spremanje jedinki u XML zapis. Također je jedinke moguće učitati iz XML zapisa kako bi ih se naknadno primijenilo na nove primjere.

5.1.1 Operatori

Svi operatori su binarni, odnosno imaju dva operanda pa tako i dva čvora-djece. Postoje četiri vrste operatora: zbrajanje, oduzimanje, množenje i dijeljenje. Ako su operandi varijable ili konstante njihove vrijednosti se direktno uzimaju. U slučaju da je neki od operanda ponovo operator, njegova vrijednost se računa rekurzivno, tako da je u teoriji moguće beskonačno duboko povezivati operatore.

5.1.2 Varijable

Varijable služe za učitavanje parametara ruba kojeg ocjenjujemo kao i njemu susjednih rubova. Svaki čvor koji je varijabla se sastoji od dva dijela. Prvi dio određuje iz kojeg



Slika 26: primjer jednostavnog genotipa

ruba, s obzirom na trenutni rub, se varijabla treba očitati. Ovo se postiže spremanjem niza koraka po polu-rubovima koje je potrebno napraviti kako bi se došlo do tog ruba. Postoje samo dvije vrste koraka, a to su F ("forward") koji skače na sljedeći polu rub u trenutnom trokutu i A ("across") koji prelazi na polu-rub koji je susjedan trenutnom polu-rubu. Ove dvije jednostavne akcije pružaju dovoljno slobode za referenciranje bilo kojeg ruba koji je direktno ili indirektno povezan s trenutnim rubom. Drugi dio čvora određuje tip vrijednosti kojeg je potrebno pročitati, postoje tri vrijednosti.

- Površina - površina trokuta kojima zadani rub pripada
- Broj susjeda - broj vrhova susjeda završnom vrhu ruba
- Kut - kut između trokuta kojima zadani rub pripada

5.1.3 Konstante

Konstante su jednostavni čvorovi koji sadrže jedan konstantan broj koji se nasumično generira u zadanom rasponu.

5.2 Križanje

Prilikom križanja odabiru se dva nasumična operatora iz stabla obje jedinke. Lijeva grana nasumičnog čvora izabranog iz prve jedinke se zamjenjuje desnom granom na-

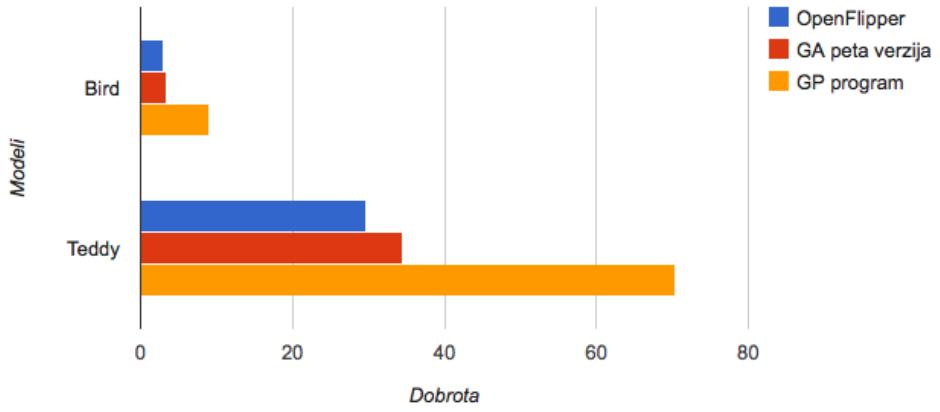
sumičnog čvora odabranog iz druge jedinke. Tako dobivena jedinka prolazi provjeru visine, i ako je unutar zadanih granica prihvaća se kao dijete.

5.3 Evolucija

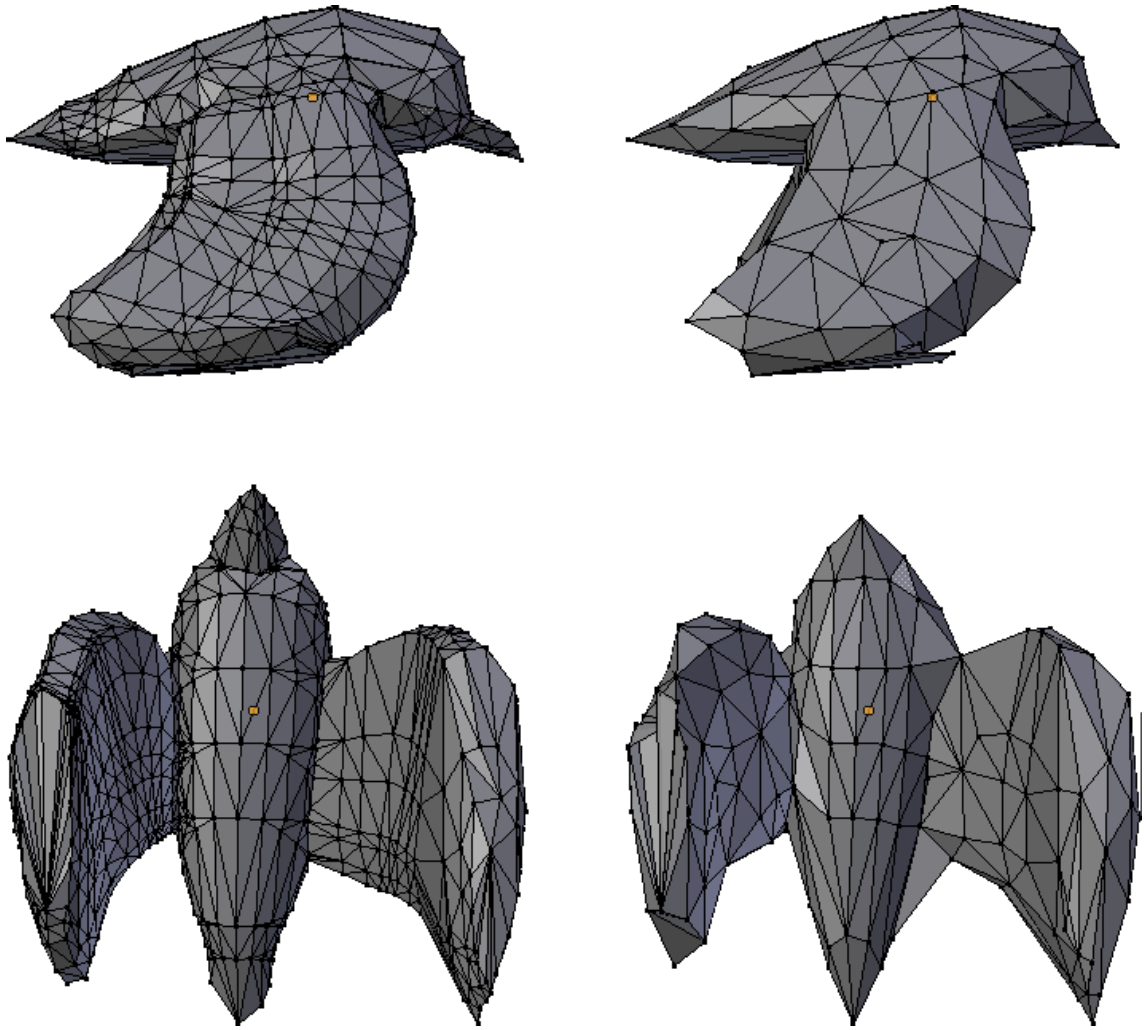
Početna generacija se stvara nasumično, uz kontrolu visine stabla i odbacivanje svih jedinki koje premašuju ograničenja. U svakom koraku od trenutne populacije se stvara zadani broj djece koja se nakon stvaranja dodaju u populaciju, jedinke se evaluiraju i sortiraju po dobroti te se odbacuje zadani broj najlošijih jedinki. Prilikom biranja jedinki za stvaranje djeteta odabiru se 3 nasumične jedinke, najgora se odbacuje a preostale dvije postaju roditelji.

5.4 Rezultati

Pojednostavljeni modeli dobiveni programima koje je ovaj postupak stvorio zadržali su svoje osnovne osobine, no ipak rezultati su vidno lošiji od rezultata postojećih metoda. Primarni razlog za to je što su generirani programi relativno jednostavni, koristi se isključivo metoda koja procjenjuje dobrotu ruba te se rubovi slijedno urušavaju s obzirom na njihovu dobrotu. Programu nedostaju metode koje bi detektirale uvjete zaustavljanja, odnosno situacije u kojima se rubovi ne smiju urušiti bez obzira na njihovu dobrotu, kao i metode koje bi s obzirom na različita svojstva rubova koristila različite metode za mjerenje dobrote. Također iako je prikupljanje podataka o modelu vrlo fleksibilno, generirani programi nemaju mogućnost iteriranja po svim rubovima nekog vrha što je bitan način kretanja ali koji na žalost zahtjeva korištenje petlji unutar programa, što predstavlja zahtjevan problem u genetskom programiranju. Nakon detaljnijeg upoznavanja s problemom zaključeno je kako zbog svojih svojstava on nije pogodan problem za rješavanje postupkom genetskog programiranja, problem je još uvijek dovoljno jednostavan da ga je čovjeku moguće analizirati i oblikovati programsko rješenje s obzirom na donesene zaključke. Automatski postupci su pogodni za podešavanje parametara programa ili rješavanja specifičnih pod-problema. U tablici 5.1. su navedeni primjeri rezultata koji su vidljivi i na slikama 27, 28 i 29.



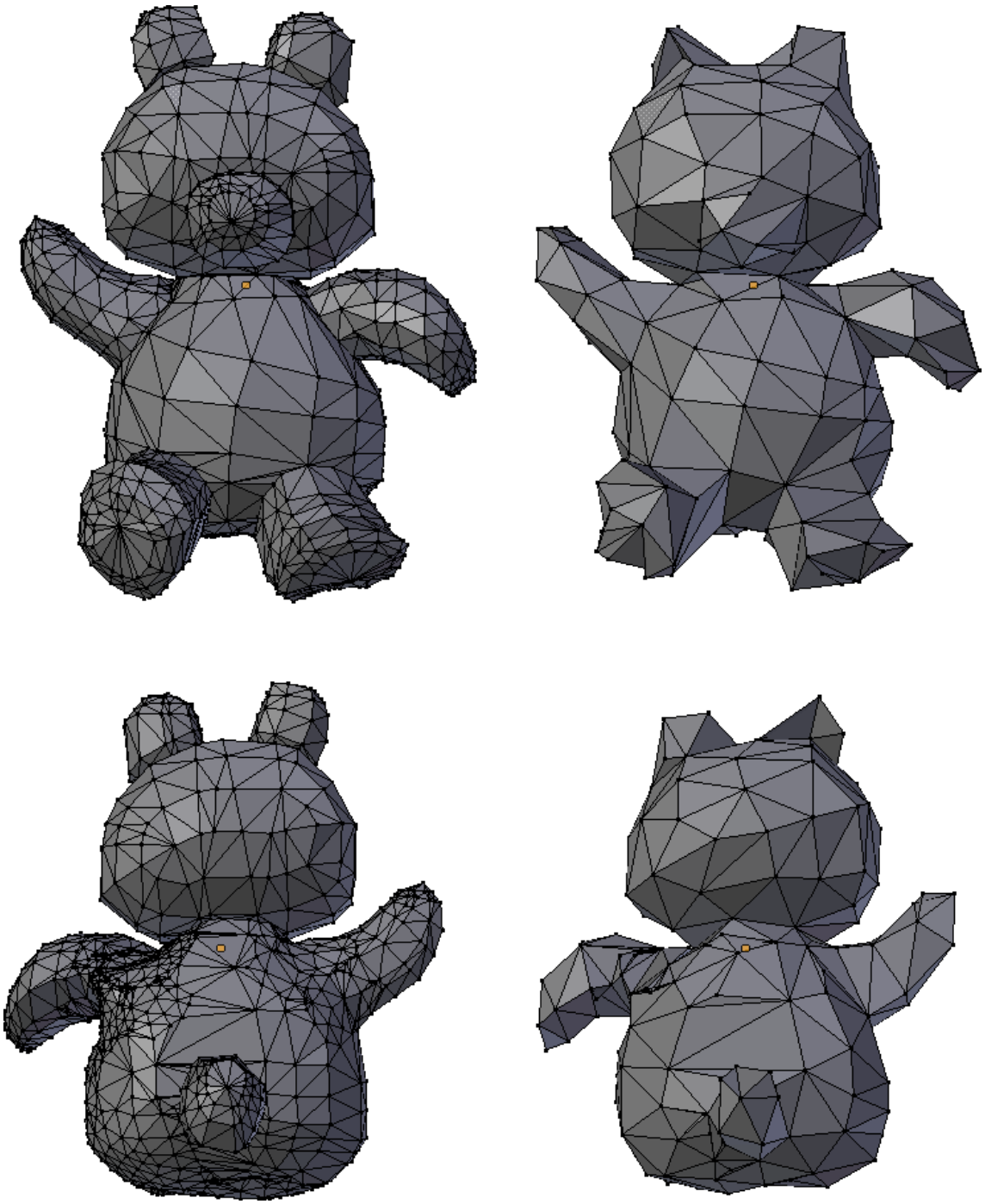
Slika 27: rezultati dobiveni optimizacijom modela genetskim programiranjem



Slika 28: originalni model (lijeva strana) i primjer rezultata dobivenog pomoću GP (desna strana)

Model	Broj trokuta	Broj trokuta	OpenFlipper	GA peta inačica	GP program
bird	1394	400	2.929	3.401	9.060
teddy	3192	500	29.604	34.406	70.436

Tablica 5.1: rezultati dobiveni optimizacijom modela genetskim programiranjem



Slika 29: originalni model (lijeva strana) i primjer rezultata dobivenog pomoću GP (desna strana)

6 Zaključak

Unutar rada je isproban veći broj potencijalnih primjena evolucijskih algoritama na problem optimizacije 3d modela uklanjanjem poligona. Rezultati znatno variraju za različite metode ali čak ni najbolji rezultati nisu uspjeli nadmašiti već postojeće algoritme. Korištenje genetskog algoritma za popunjavanje rupa nastalih uklanjanjem vrhova se pokazalo najboljom metodom i uz dodatnu optimizaciju parametara bi potencijalno mogla dati vrlo kvalitetna rješenja. Međutim, upitno je koliko bi taj postupak bio isplativ zbog relativno loših performansi genetskog algoritma.

7 Literatura

1. L. Kobbelt, S. Campagna, H. Seide; A General Framework for Mesh Decimation; University of Erlangen-Nurnberg

2. A. Ciampalini, P. Cignoni, C. Montani, R. Scopi; Multiresolution Decimation based on Global Error; Istituto per l'Elaborazione dell'Informazione; 1997.

3. P. Cignoni, C. Montani, R. Scopigno; A comparison of mesh simplification algorithms; 1997.

4. H. Hoppe; Progressive Meshes; Microsoft Research

5. OpenMesh 2.2 Documentation; http://openmesh.org/Documentation/OpenMesh-2.2-Documentation/mesh_hds.html

6. W. J. Schroeder, J. A. Zarge, W. E. Lorensen; Decimation of Triangle Meshes; General Electric Company

7. D. Mount; Geometric Data Structures for Games: Meshes and Manifolds; 2013.

Optimizacija 3D modela smanjenjem broja poligona uz pomoć evolucijskih algoritama

Unutar ovog rada se istražuju mogućnosti korištenja evolucijskih algoritama, primarno genetskih algoritama i genetskog programiranja, na problemu optimizacije 3D modela uklanjanjem poligona. Osnova problema je odabrati i ukloniti poligone koji će minimalno deformirati objekt, odnosno od originalnog modela dobiti pojednostavljeni s manjim brojem poligona, ali što vjerniji originalu.

Ključne riječi: 3d modeli, genetski algoritmi, genetsko programiranje, evolucijski algoritmi

Optimization of 3D models by reducing polygon count using evolutionary algorithms

This paper covers using evolutionary algorithms, primary genetic algorithms and genetic programming, on a mesh simplification problem. Core of the problem is picking and removing polygons from a model while keeping it as similar to the original as possible.

Keywords: 3d models, genetic algorithms, genetic programming, evolutionary algorithms