SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5392

DETEKCIJA JEZGRI STANICA UZ POMOĆ ALGORITAMA EVOLUCIJSKOG RAČUNANJA

Josip Matak

Zagreb, lipanj 2018.

ZAHVALA

Najveća zahvalu dugujem mojoj obitelji koja mi je bila najveći poticaj i uzor tijekom cijelog dosadašnjeg školovanja. Posebno HVALA ide mojim roditeljima koji su bezuvjetno uvijek bili tu za mene kada mi je bilo najpotrebnije, a takvih trenutaka bilo je mnogo.

Zahvaljujem se mentoru prof. dr. sc. Domagoju Jakoboviću na strpljivosti, savjetima i podršci kako u pisanju ovog rada tako i u cjelokupnom dosadašnjem mentorstvu.

Sadržaj

1. Uvod

Medicinska i farmaceutska industrija već duže vremena s razlogom su jedne od najbrže rastućih, a ujedno i najunosnijih industrija. Razvoj novih lijekova je sve skuplji. Enroomov zakon [1] kaže da se broj lijekova koji je moguće dobiti za milijardu dolara svake godine smanjuje logaritamski. Razvojem računala i tehnologije taj proces se pokušava ubrzati. Cilj ovog rada je obraditi metodu za analiziranje mikroskopske slike stanica.

Stanica je posebno važan dio u farmaceutskom istraživanju zato što njezina jezgra nosi potpunu uputu DNA koju je moguće analizirati i doći do zaključaka o djelovanju određenih kemikalija na neku vrstu tkiva.

Obrada mikroskopske slike samo je jedna u nizu metoda kojom se znanstvenicima pokušava olakšati posao analize i proizvodnje lijekova. Sama izrada dugotrajan je i mukotrpan proces. Jedan od koraka koji je većim dijelom automatiziran je postavljanje pokusa. Roboti zaduženi za pripremu automatski izdvajaju milijune stanica tkiva, spremaju njihove slike i predaju im različite kemikalije. Različite vrste bolesti zahtijevaju različite tretmane analize stanica nakon pokusa, primjerice, kod nekih je potrebno promatrati promjenu boje dok je kod drugih moguće primijetiti promjenu oblika.

Trenutno usko grlo događa se pri analizi mikroskopske slike stanica koja se dobiva nakon pokusa. Znanstvenici su zaduženi za ručno pregledavanje tisuća slika što lako vodi ka pogreškama. Kao prvi korak navodi se izdvajanje jezgri stanica iz slike. Slike se primaju u različitim oblicima i različitih boja zbog čega je teško jasno odrediti gdje je stanična jezgra.

Trenutno razvijeni modeli izdvajaju jezgre koje su točno određenog oblika ili boja, i to je razlog zašto je korisno pokušati pronaći računalni model evolucijskim algoritmom koji bi mogao precizno odrediti položaj stanice i znanstvenicima ubrzati rad na izradi novih lijekova.

U početnom dijelu rada obrađen je teorijski dio algoritama evolucijskog računanja, odnosno genetskog algoritma i genetskog programiranja. Zatim su opisani alati korišteni u obradi slike.

Ta dva elementa služe kao podloga nastavka u kojem su navedene tehnike poslužile za izradu programa koji se bavi segmentacijom jezgri stanica iz danih mikroskopskih slika.

Zaključni dio rada daje uvid u postignute rezultate i još jednom promiče vrijednost bavljenja biomedicinskim istraživanjima unutar računarske znanosti.

2. Algoritmi evolucijskog računanja

Problemi koji se svode na pretraživanje ogromnog prostora stanja danas su sve aktualniji i sve češće imaju korisnu primjenu u industriji. Priroda takvog problema, nedostatak vremena i računalne snage koja bi mogla obraditi sva rješenja naveli su znanstvenike da izmisle metode kojima će se ti problemi svesti na jednostavnije te omogućiti pronalazak zadovoljavajućeg rješenja.

Kao što im ime navodi, algoritmi evolucijskog računanja su biološki inspirirani algoritmi nastali kao odgovor na probleme optimizacije. Simulacijom prirodnog procesa evolucije kojeg je prvi puta predstavio britanski znanstvenik Charles Darwin, nastoji se iz generacije u generaciju prenijeti najbolji genetski materijal te iz njega stvoriti jedinke koje zadovoljavaju potrebe našeg problema. Takvim pristupom iz gomile nasumičnih jedinki kroz prirodnu selekciju, križanje i mutaciju moguće je usmjeriti pretraživanje prostora stanja prema najboljim rješenjima.

2.1. Genetski algoritam

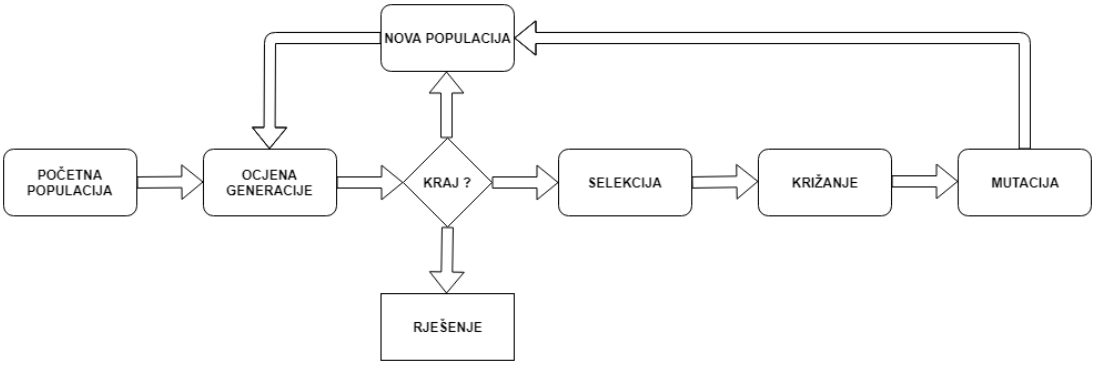
Simulacija prirodnog evolucijskog ponašanja započela je još 1954. radom Nilsa Aalla Baricellija [2]. Njegov rad bio je usredotočen na učenje igranja igara, no kasnije je genetski algoritam iskorišten kao moćan alat pri raznovrsnim problemima optimizacije i umjetnoj inteligenciji.

Genetski algoritam možda je i najbolji, školski primjer evolucijskog računanja. Kod njega je prirodni proces evolucije potpuno preslikan i jasno istaknut. Baš iz tog razloga algoritam je fleksibilan i lagano uporabljiv nad širokim spektrom problema. Njegova izrazito pozitivna strana je jednostavnost implementacije što omogućuje korisniku da se usredotoči na stvari koje se vežu teorijske podloge problema, a manje same implementacijske logike. Premda se radi o stohastičkom algoritmu (ne jamči pronalaženje najboljeg rješenja), rješenja koja se pronađu u praksi su uglavnom zadovoljavajuća u kontekstu problema koji se rješava.

2.1.1. Opis algoritma

Algoritam je konstruiran na način da je prvi korak generiranje nasumičnog skupa rješenja koji se naziva populacija. Svako rješenje predstavlja jednu jedinku čime možemo povući paralelu s prirodnom evolucijom. U prvom koraku proizvedeni rezultati su uglavnom loši i taj korak služi inicijalizaciji. Da bismo odredili koliko je jedinka prilagodljiva odnosno koliko je proizvedeno rješenje ujedno i dobro moramo imati način ocjenjivanja jedinki. Takvu funkciju, kojom iz predanog rješenja dobijemo ocjenu prilagodbe nazivamo funkcijom „dobrote“ (engl. *Fitness function*).

Nakon postavki temelja algoritma, događa se proces evolucije, što je vidljivo na Slici 1. Iz početne populacije određenom metodom selekcije biraju se jedinke koje će s nekom vjerojatnošću ući u proces križanja ili bez njega napredovati u sljedeću generaciju. U procesu križanja dvije roditeljske jedinke stvorit će dijete koje će sadržavati genotip oba roditelja te će proći proces mutacije koji će omogućiti da novi genetski materijal uđe u populaciju s tom nasumičnom i nepredvidivom promjenom. Nova generacija, ovisno o elitizmu (svojstvo kojim najbolje jedinke prethodne generacije nepromijenjene napreduju u novu) sastojat će se od novih jedinki stvorenih križanjem i mutacijom starih te najboljim jedinkama iz prethodne generacije. Time će se zadržati sve ono dobro što je do sada pronađeno.



Slika 1 Dijagram toka evolucijskog procesa

2.1.2. Selekcija

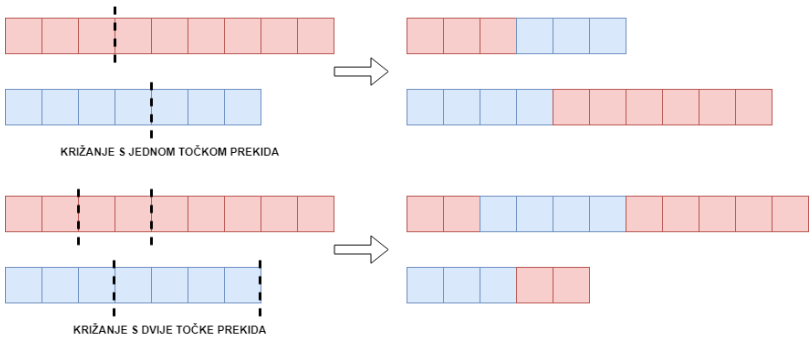
Jednu jedinku iz populacije moguće je odabrati na razne načine. U ovom radu korištene su dvije metode uzimanja jedinke:

- **TURNIRSKA SELEKCIJA** – nasumično je odabrano *n* jedinki iz populacije, zatim se između njih preuzima ona s najboljom vrijednošću dobrote.
- **PROPORCIONALNA SELEKCIJA** (engl. *roulette-wheel selection*) – populacija se poput kotača za rulet podijeli na dijelove, međutim svaki dio je onoliko velik kolika je proporcija između dobrote te jedinke i normalizirane sume svih ostalih jedinki. Vjerojatnost uzimanja bolje jedinke je veća od vjerojatnosti uzimanja one lošije.

2.1.3. Križanje

Križanje je bitan faktor u progresu evolucijskog algoritma. Želja je iz roditelja izgraditi tim bolju jedinku djeteta za sljedeću generaciju. To možemo postići različitim metodama:

- **KRIŽANJE S PREKIDOM U JEDNOJ TOČKI** – unutar jedinke se postavi jedna točka prekida, nakon toga slijedi zamjena genetskog materijala između dvije roditeljske jedinke
- **KRIŽANJE S PREKIDOM U DVIJE TOČKE** - svaka jedinka presječena je s dvije točke presjeka, rezultat je spajanje zasebnih dijelova iz obje jedinke



Slika 2 Prikaz križanja

2.1.4. Mutacija

Svrha mutacije je unos novog genetskog materijala unutar populacije. Katkad je dobro izići izvan dosad posjećenog prostora stanja pretraživanja i mutacijom prijeći u neki neistraženi dio. Unutar genetskog algoritma ovog rada korištena je samo jedna vrsta mutacije:

- **JEDNOLIKA MUTACIJA** – svaki gen iz genotipa ima određenu vjerojatnost da bude mutiran, ukoliko se mutacija dogodi, gen je zamijenjen s njemu prikladnim genom iz skupa svih mogućih

2.2. Genetsko programiranje

Do sada opisani principi evolucijskog računanja primjenjuju se i na genetsko programiranje. Jedina velika razlika kod GP-a je prikaz jedinke. Kod genetskog programiranja jedinka je stablo i nad njom se primjenjuju različite tehnike mutacije i križanja.

2.2.1. Opis algoritma

Jedno rješenje problema predstavljeno je u stablastoj strukturi koja je sačinjena od završnih (terminalnih) i nezavršnih čvorova. Struktura se može sagledati kao program u izvođenju gdje će, ovisno o ulaznim parametrima biti izvršeni različiti dijelovi stabla. Nezavršni čvorovi će uglavnom biti različite funkcije koje će s obzirom na ulaz izvršavati neku granu stabla dok će završni predstavljati neku akciju koja treba biti obavljena nad ulazom u stablo.

2.2.2. Inicijalizacija

Početna populacija može se generirati na nekoliko različitih tehnika:

- **METODA *GROW*** – nasumično se generiraju čvorovi iz operatorskih i terminalnih skupova čvorova, taj postupak se ponavlja dok nismo generirali sve terminalne čvorove ili dosegli maksimalnu dubinu stabla

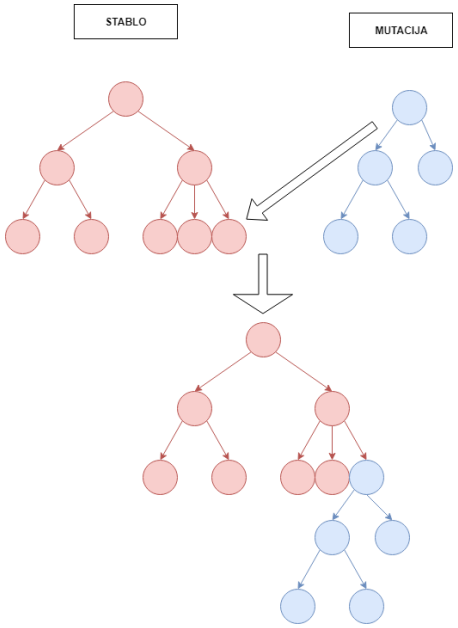
- **METODA *FULL*** – do maksimalne razine generiraju se nezavršni čvorovi, kada smo postigli maksimalnu dubini generira se terminalni čvor. Ovakvom proizvodnjom stabla uvijek će se generirati potpuno stablo.
- **METODA *RAMPED HALF-AND-HALF*** – populacija se podijeli na jednake dijelove. Polovica svakog dijela generira se metodom *grow*, a druga metodom *full.* Redom za svaki od jednakih dijelova generiraju se stabla za jednu dubinu veća.

2.2.3. Križanje

Metoda križanja stablastih jedinki u ovom slučaju je prilično jednostavna. Unutar stabla odabran je jedan čvor križanja te se u njega ubacuje čvor iz drugog stabla. Isti princip je ponovljen za drugu jedinku, a između dvoje generirane djece uzima se ono bolje.

2.2.4. Mutacija

Mutacija je izvedena na sličan način poput križanja. Unutar stabla nasumično je izabran jedan čvor i u njega je ubačen korijen nasumično generiranog stabla jednom od metoda generacije što je prikazano na Slici 3.



Slika 3 Mutacija kod genetskog programiranja

3. Obrada slike

Za obradu slike, većim dijelom ovog rada korišteni su algoritmi pretprocesiranja kojima je cilj poboljšati početni oblik slike, nakon čega slijedi segmentacija slike s ciljem da se pronađu početno traženi elementi na slici.

Na računalu, slika je učitana kao trodimenzionalna matrica pri kojoj je na svakoj poziciji originalnog piksela spremljena odgovarajuća boja prikazana kao trodimenzionalni vektor. Ukoliko je to potrebno u algoritmu, zadana boja mijenja se odgovarajućim intenzitetom sive boje, a slika postaje crno-bijela.

3.1. Morfološke transformacije

Prva vrsta procesiranja slika je obrada takozvanim morfološkim transformacijama. To su jednostavne operacije koje kao ulaze uzimaju jednu sliku i matricu transformacije (nije povezana s matricama primjenjivanim u interaktivnoj grafici), koju nazivamo strukturnim elementom.

Taj strukturni element zapravo je konvolucijski element koji je zajedno s originalnom slikom iskorišten za izračun konačnog rezultata. To bi značilo da za svaki element matrice slike oko njega postavljamo konvolucijsku matricu, a potom se posebnom metodom izračunava njegova nova vrijednost.

Vrste morfoloških transformacija primjenjivanih u radu dane su u nastavku, gdje ***A*** predstavlja matricu slike, a ***B*** matricu strukturnog elementa:

a) **DILATACIJA** – konačni cilj ove transformacije je proširiti elemente slike sa strukturnim elementom

$$A \oplus B = \{z \mid (B)_z \cap A \neq \emptyset\}$$

Rezultat je skup svih *z* za koje vrijedi da ukoliko reflektiramo ***B*** oko centra, element *z* ima presjek s ***A***

b) **EROZIJA** – transformacija koja ima cilj suprotan diletaciji, suziti elemente početne slike

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

Konačno rješenje je podskup matrice ***A*** za sve *z* koje čini refleksija matrice ***B***.

c) **OTVARANJE** – Kombinacija je redom diletacije i erozije, služi kako bi se ublažila kontura objekta, ukinule tanke veze između objekata i eliminirali mali oštri dijelovi

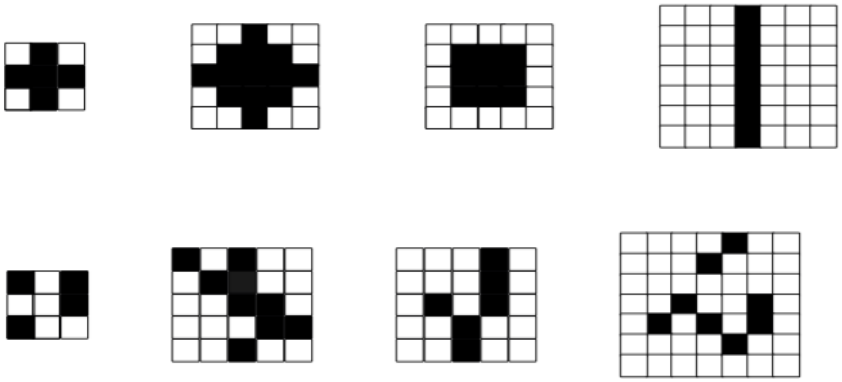
$$A \circ B = (A \ominus B) \oplus B$$

d) **ZATVARANJE** – Primjenjuje se erozija, a zatim diletacija. Također, poput otvaranja, ublažava konture objekta, s druge strane stapa uske prekide i duge tanke uvale, a i eliminira male rupe.

$$A \circ B = (A \oplus B) \ominus B$$

3.2. Strukturni elementi

Unutar programa upotrebljavani su takozvani „regularni“ strukturni elementi koji se sačinjavaju od 2D matrice popunjene nulama i jedinicama, a prirodnog su oblika (linija, kvadrat, križ, dijamant) [3]. Oni su primjenjivi na binarnu sliku, tj sliku koja je učitana u crno-bijelom formatu. Takve matrice moraju biti kvadratne matrice kako bi mogle biti primjenjive na sliku. S druge strane „neregularni“ strukturni elementi također slijede uporabu kvadratne matrice, ali ovaj puta elementi unutar su nasumično poredani, što otvara mogućnost za njihovu lakšu promjenu. Primjer obje vrste strukturnih elemenata je prikazan u nastavku:



Slika 4 Regularni elementi (gore), neregularni (dolje)

3.3. Određivanje praga

Prilikom izdvajanja elemenata slike koje je potrebno prepoznati koristi se metoda određivanja praga. Nakon što je odrađena transformacija, ili neki određeni niz transformacija, elemente koji su dobiveni potrebno je odsjeći u one koji pripadaju skupu kojeg prepoznajemo i one koji ne pripadaju. Za to se koristi metoda određivanja i odsijecanja praga. Takve metode rade na način da iskoriste određenu matematičku funkciju kojom će dobiti prag na neki posebno određen i željen način, a potom će sve elemente iznad toga praga svrstati u jednu kategoriju, a elemente ispod praga u drugu kategoriju. U ovom radu bit će opisana metoda određivanja praga koju je predložio japanski matematičar Nobuyuki Otsu [3].

3.3.1. Otsuova metoda

Pri Otsuovoj metodi, iscrpno se pretražuje prag koji minimizira varijancu unutar razreda podjele, definiranu kao težinska suma varijanci dviju klasa i sume vjerojatnosti u klasi. Na početku je izračunat histogram vjerojatnosti pojavljivanja određenog intenziteta boje na slici. Veliko ograničenje ovog algoritma je očekivanje da će histogram biti bimodalni, što je primjenjivo na slikama u kojima je izražena svijetla i tamna strana:

$$\sigma_{\omega}^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

,

gdje ω_0 predstavlja sumu vjerojatnosti do praga, a ω_1 sumu vjerojatnosti nakon praga:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i) \; ; \; \omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Izgled algoritma možemo zapisati na sljedeći način:

1. Izracunaj histogram i vjerojatnosti za svaki intezitet
2. Postavi inicijalne $\omega_i(0)$
3. Za svaki $t \in (1, \max(intenzitet))$:
    1. Ažuriraj ω_i
    2. Izračunaj $\sigma_\omega^2(t)$
4. Vrati t koji odgovara min $\sigma_\omega^2(t)$

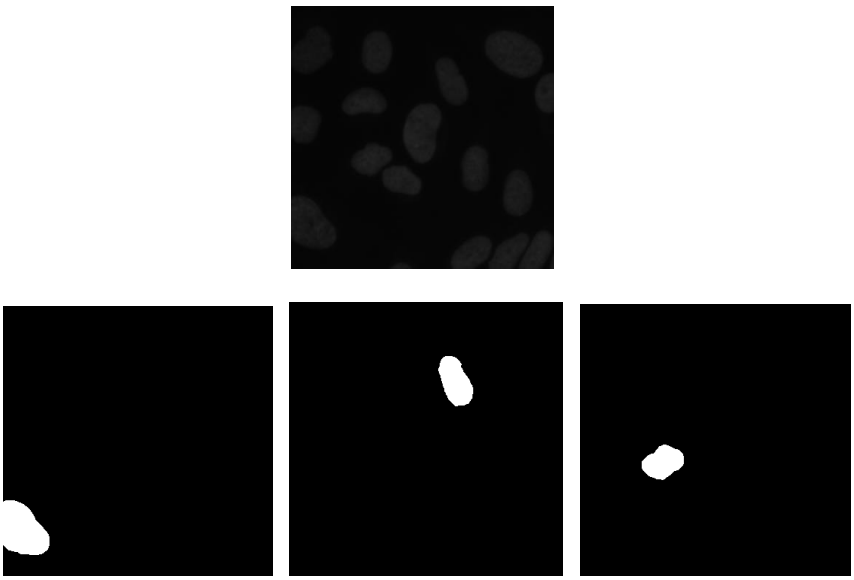
Slika 5 Otsuov algoritam

Ovakav algoritam sklon je pogreškama pri malim promjenama varijacije intenziteta unutar slike uzrokovanim šumom, višim intenzitetom pozadine i velikim objektima unutar slike. Razlog tomu je što se prag izračunava temeljem cijele slike, na globalnom području. Primjer rada algoritma prikazan je na slici:

Slika 6 Otsuova metoda primijenjena na primjeru segmentacije kovanica

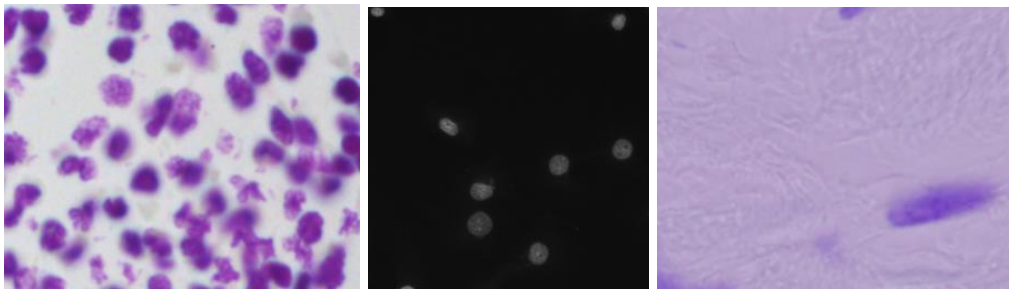
4. Detekcija jezgri stanica

Gore navedeni algoritmi analizirani su na jednoj posve zanimljivoj primjeni, detekcija jezgri stanica. Ulaz u program je veliki skup mikroskopskih slika stanica, te za svaku sliku analogni skup maski koje označavaju poziciju jezgre unutar slike. Podaci su preuzeti sa stranice *„Kaggle“* u sklopu znanstvenog natjecanja u 2018. godini [4].



Slika 7 Primjer ulazne mikroskopske slike (gore) i nekoliko detektiranih stanica (dolje)

Zadani skup slika sastoji se od slika različitih dimenzija, stanica različitih oblika i vrsta, i pozadinskih tkiva različitih boja. Takve značajke uvelike otežavaju detektiranje elemenata koje je potrebno izdvojiti iz slike. Primjer različitosti među slikama dan je u nastavku:

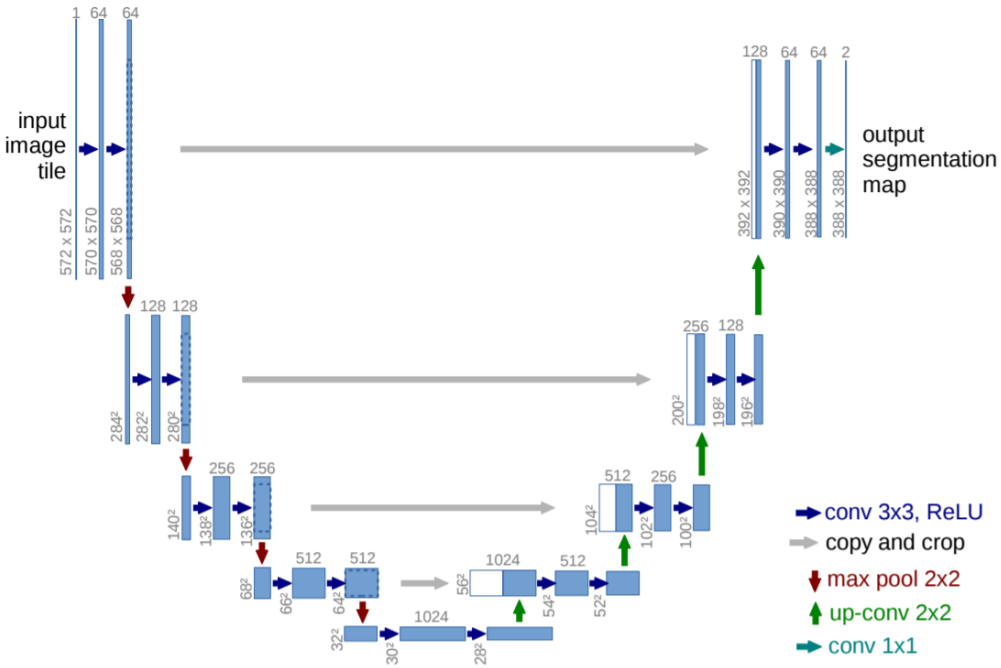


Slika 8 Vrste mikroskopskih slika

4.1. Uobičajeni postupci

Općeniti postupak koji bi se proveo kod ovakvog problema najčešće obuhvaća matematički model koji bi omogućio prilagodbu u ovisnosti o tome kako izgleda skup slika za učenje modela. Takvo ponašanje nam omogućuju neuronske mreže. Za biomedicinsku segmentaciju razvijena je takozvana U-NET arhitektura neuronske mreže [5]. Posebna je to vrsta konvolucijskih neuronskih mreža, koje neće biti opisane u ovom radu.

Zahtjev koji se postavlja na takve metode jest taj da svaka slika ima predefiniranu veličinu koja će odgovarati ulazima u neuronsku mrežu. Naziv U-NET arhitektura dolazi iz njezina izgleda prikazanog na Slici 9.



Slika 9 U-net arhitektura. Svaki plavi kvadrat označava kanal kroz koji prolazi ulaz. Strelice označavaju različite operacije i navedene su pored.

Mana ovog pristupa je manjak robusnosti na promjene. Promjenom veličina, raznovrsnošću objekata namijenjenih segmentaciji, mreža sve teže i teže savladava prepreke stavljene pred nju. Ipak, ova posebna vrsta mreža i načina segmentacije našla je svoju upotrebu, i ima visoku prilagodljivost na primjene iz nekog užeg skupa raznolikosti.

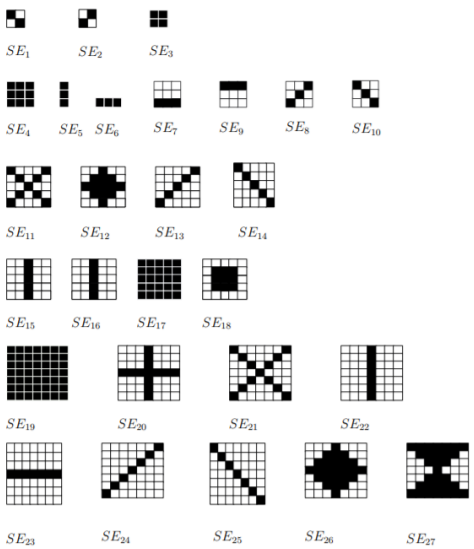
4.2. Opis modela

U ovom radu obrađen je jedan potpuno drugačiji pristup. Ideja koja se krije iza pristupa je omogućiti veću robusnost nad dosadašnjim tehnikama. Teza koja je postavljena je može li se pronaći niz morfoloških transformacija nakon čije primjene i odsijecanjem praga možemo dobiti željeni segmentirani dio slike.

Jedna transformacija slike bit će oblikovana način da se nad slikom primjeni određena vrsta morfološke transformacije uz određeni strukturni element.

Strukturni elementi u programu zadani su na dva načina. Prvi način je predefiniranih 27 strukturnih elemenata koji su svi regularni, prikazani su na Slici 10. Drugi način definira veličinu kvadratne matrice, a jedinice i nule su posložene nasumično, kako bi se omogućilo programu da sam pronađe zadovoljavajući strukturni element.

Kod prvog slučaja u kod su predefinirani zadani strukturni elementi, zapisani ručno, dok u drugom slučaju strukturni element određuje objekt čiji prvi element označava veličinu matrice, a drugi element raspodjelu jedinica i nula.



Slika 10 Regularni, predefinirani strukturni elementi [3] u programu

4.2.1. Genetski algoritam pri segmentaciji slike

Za određivanje redoslijeda primjene transformacija na sliku uzet je genetski algoritam. Njegov konačni cilj pronaći je što bolji redoslijed koji bi dao rezultate primjenjivane na skup slika koje su korištene u treniranju, kao i na skup slika koje dotada nisu viđene.

Jedinka genetskog algoritma je lista parova morfoloških transformacija i pripadajućeg strukturnog elementa. Prikaz jedne jedinke je u nastavku:

$$\mathbf{ind = [\ (D,\ SE_4),\ (O,\ SE_{12}),\ (D,\ SE_{11})\ ]}$$

Primjer označava da će se redom izvoditi operacije diletacije, otvaranja a potom ponovno diletacije, zajedno sa pripadajućim indeksima regularnih strukturnih elemenata.

Takvim načinom, prostor pretraživanja može biti širokog opsega i nemoguće ga je cijelog proći. Ukoliko je maksimalna veličina liste zadana s *n*, a kardinalitet skupa definiranih transformacija i matrica zadani s *t* i *m* respektivno, onda je veličina prostora pretraživanja dana sa:

$$\sum_{i=0}^{n} (t \cdot m)^i = \frac{(m \cdot t)^{n+1} - 1}{(m \cdot t) - 1},$$

što bi u slučaju s 5 morfoloških transformacija, 27 strukturnih elemenata i maksimalnom veličinom liste od 20 elemenata bilo preslikano u:

$$t = 7, m = 27, n = 20$$

$$\frac{135^{21} - 1}{189 - 1} = 3.4 \cdot 10^{45}$$

U inačici s neregularnim strukturnim elementima taj broj se višestruko povećava.

Na početku je generirana nasumična populacija jedinki genetskog algoritma. Ta populacija sastoji se od listi čije su veličine nasumični brojevi

između 1 i predefiniranog *max_ind*, maksimalne veličine početno generirane jedinke. Nakon toga, generirane jedinke su poslane na evaluaciju. Evaluacija je složen proces i zahtjeva veliko vremensko opterećenje, u ovisnosti je s brojem jedinki u populaciji i brojem slika koji su iz početnog skupa uzete kao slike za učenje. Taj broj zadan je kao parametar na početku programa.

4.2.1.1. Određivanje dobrote jedinke

Na temelju dobivene liste transformacija potrebno je odrediti njenu dobrotu. Dobrota će biti određena na različite načine, a svaki od njih rezultirat će brojem od 0 do 1, gdje jedinke s dobrotom bliže jedan označavaju one bolje.

Kao što je već rečeno, svaka slika ima niz analogno prepoznatih jezgri stanica u obliku maske. Kako bi mogli procijeniti koliko dobro program prepoznaje elemente slike, maske je potrebno spojiti u jednu sliku koja će biti uspoređivana s procesuiranom slikom. Potom, da program ne bi ovisio samo o broju pogođenih piksela, dobrota je prilagođena da predviđa i broj prepoznatih jezgri na slici.

Za početak, potrebno je objasniti način na koji se uspoređuju dvije slike, one procesuirane podijeljene s prethodno određenim pragom, i one koja nam je dana kao ulaz za učenje.

Pri tome će nam poslužiti takozvana „matrica zbunjenosti“ dana u nastavku:

		Prava vrijednost	
		Bijeli piksel	Crni piksel
Predviđena vrijednost	Bijeli piksel	TP (true positive)	FP(false positive)
	Crni piksel	FN (false negative)	TN(true negative)

Tablica 1 Matrica zbunjenosti

U ovisnosti o tomu kakav rezultat je izlaz programa, vrijednosti predviđanja bit će izračunate. S obzirom na način baratanja vrijednostima odabrana su tri pristupa pomoću kojih je moguće dobiti dobrotu izlaza:

- **JEDNOSTAVNA USPOREDBA** – računa se kao omjer precizno prepoznatih piksela i ukupnog broja piksela.

$$f(image, mask) = \frac{TP + TN}{TP + TN + FP + FN}$$

Ovakav način je najjednostavniji i ima nekoliko mana koje je potrebno popraviti u nastavku.

- **UNAPRIJEĐENA JEDNOSTAVNA USPOREDBA** – razlika kod ove i jednostavne usporedbe je ta što se u konačan rezultat ne ovisi o točnim negativnim pikselima. Razlog da bi se nešto takvo napravilo je taj što su precizni pozitivni pikseli daleko vrjedniji i u manjoj količini, stoga ovakav način vrednovanja producira relevantnije rezultate.

$$f(image, mask) = \frac{TP}{TP + FP + FN}$$

- **TEŽINSKA USPOREDBA** – I dalje producira slične vrijednosti, no svaki je parametar matrice zbunjenosti pomnožen s težinskim faktorom koji u sumi određuje koliko je taj element doista važan za konačni rezultat.

$$f(image, mask) = \frac{w_{TP} \cdot TP + w_{TN} \cdot TN}{w_{TP} \cdot TP + w_{TN} \cdot TN + w_{FP} \cdot FP + w_{FN} \cdot FN}$$

Sljedeća stavka o kojoj treba ovisiti rješenje je broj predviđenih jezgara stanica. Pri početnom učitavanju taj podatak nam je poznat, i program je to bolji što bolje pogađa rezultat. Nagrađuje se preciznost prema rješenju. Stoga funkcija kojom se određuje dobrota broja jezgara zadana je na sljedeći način:

$$f(image,k) = \begin{cases} \frac{p}{k}, & p < k \\ 2 - \frac{p}{k}, & k < p < 2k \\ 0, & p > 2k \end{cases},$$

gdje *p* predstavlja broj jezgri stanica nakon procesiranja slike, a *k* pravi broj jezgri stanica na slici.

Lako je uočiti da ovakav pristup ima veliku manu, ne možemo precizno odrediti radi li se o točnim mjestima stanica ili su to izolirana područja nastala odsijecanjem praga. Zbog toga se za određivanje dobrote koristi kombinacija ove dvije metode.

Preciznost pogođenih piksela i preciznost pogođenog broja jezgri množe se kako bi se dobila konačna dobrota jedinke. Kao primjer dana je jednostavna usporedba piksela i funkcija za određivanje preciznosti pogođenih jezgara:

$$fitness = \frac{TP + TN}{TP + TN + FP + FN} \cdot \frac{p}{k}$$

Unutar genetskog algoritma provodi se postupak koji je već opisan. Bitno je napomenuti da je algoritam nije elitistički i moguće je da se najbolja rješenja ne propagiraju dalje, iako ostaju zapisana kao najbolja u pomoćnoj strukturi.

Za sve, na određeni način selektirane jedinke iz generacije, s nekom, prije odabranom vjerojatnošću primjenjuje se operator križanja. Tim operatorom unutar generacije sačuvan je vrijedni genetski materijal iz razloga što se od nastale 2 jedinke djeteta, uzima ona bolja.

Zatim se na novonastaloj generaciji primjenjuje operator mutacije. Taj operator, kao što je navedeno, služi kako bi uveo novi genetski materijal i doveo promjenu dosadašnjeg stanja. Primjenjuje se tako da s nekom vjerojatnošću odaberemo hoće li se dogoditi, potom unutar liste transformacija u jedinki mijenjamo elemente s pripadnom vjerojatnošću. Radi se ili o promjeni vrste transformacije, ili o promjeni strukturnog elementa. Kod predefiniranih strukturnih elemenata i vrsta transformacija, jedina mogućnost zamjene je uzimanje nasumičnog elementa iz skupa mogućih elemenata, čime se unutar jedinke uzrokuje neočekivana i značajna promjena, no pokazuje se da koncept radi dobro nevezano uz takve nedostatke.

4.2.2. Genetsko programiranje pri segmentaciji slike

Drugi pristup optimiranju transformacija je primjena genetskog programiranja. Cilj je pronaći stablo koje će svojim obilaskom omogućiti pravilnu segmentaciju slike.

Koncept koji se razlikuje od genetskog algoritma je taj što izvođenje stabla ne ovisi o redoslijedu već o parametrima slike koji će biti opisani. Ocjenjivanje jedinke događa se na isti način na koji je ono opisano kod genetskog algoritma.

Stablo genetskog programiranja je po običaju podijeljeno u dvije vrste čvorova, završne i nezavršne. Završne čvorove čine transformacije oblikovane poput onih korištenih u genetskom algoritmu. Nezavršne čvorove pak čine uvjeti koji okidaju jednu od grana stabla ovisno o tome je li uvjet zadovoljen.

S obzirom na to da su nezavršni čvorovi jedina nepoznanica ovog pristupa, obrađeni su u nastavku. Glavna razlika svih slika unutar programa su njihove boje. Kako je teško brzo analizirati oblike svih objekata na slici osvrt je pao na boju kao glavni kriterij.

Za određivanje dominantne boje uzet je algoritam k-srednjih vrijednosti čiji je cilj podijeliti boje u *K* grupa. Ideja je da svaka grupa ima svoju srednju vrijednost, a svaki element pripada jednoj grupi čija mu je srednja vrijednost najbliža.

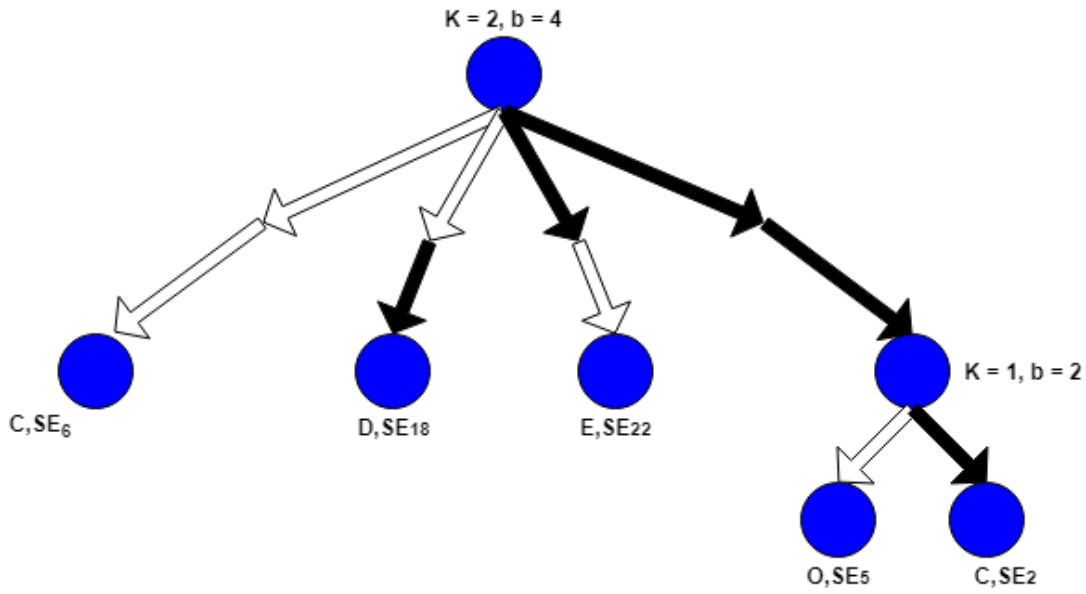
Nakon provedenog particiranja, za *K* grupa dobije se *K* srednjih vrijednosti. Za potrebe algoritma one služe kako bi na temelju ulaza (slike) odredili koji se od sljedećih *b* čvorova treba izvršiti.

Razlikuju se tri vrste nezavršnih čvorova:

1. **K = 1, b = 2**. Ovisno o tome je li intenzitet sive boje u toj jednoj grupi veći od polovice (128) ili manji okida se lijeva ili desna strana stabla.
2. **K = 1, b = 4**. Ovaj puta postoje četiri djeteta koja se mogu izvršiti jer je opseg intenziteta sive boje podijeljen na 4 skupine: [0-64],[64-128],[128-192][192-256].
3. **K = 2, b = 4**. Postoje 4 kategorije u koje može upasti. Dvije su srednje vrijednosti i za svaku je pitanje hoće li odgovarati prvoj, svjetlijoj ili drugoj, tamnijoj polovici intenziteta sive boje.

Zanimljiv je podatak da se na ovaj način izvrši samo jedan završni čvor, ali i to ograničenje se pokazalo kao prihvatljivo zbog raznolikog skupa slika i potrebe za blagom promjenom kako bi se pravilno segmentirala slika.

Na slici 11 primjer je jednog stabla nastalog unutar programa. Završne čvorove lako je prepoznati po imenima transformacija, a nezavršne po imenima funkcije kojom se vrednuje. Svijetle linije označavaju da se radi o grani koja će se odraditi ukoliko srednja vrijednost intenziteta pripada vrijednosti od 128 – 256. U korijenu stabla podjela je izvršena na dva intenziteta što je razlog stavljanja dvije strelice. Obilazak stabla za određenu sliku tekao bi na sljedeći način: u korijenu izračunate su 2 srednje vrijednosti [73][120], obje pripadaju tamnoj strani, izvršava se zadnja grana, potom je izračunata jedna srednja vrijednost [92], okida se desna grana i primjenjiva se transformacija zatvaranja s drugim strukturnim elementom.



Slika 11 Primjer jednog stabla

4.3. Korištene tehnologije

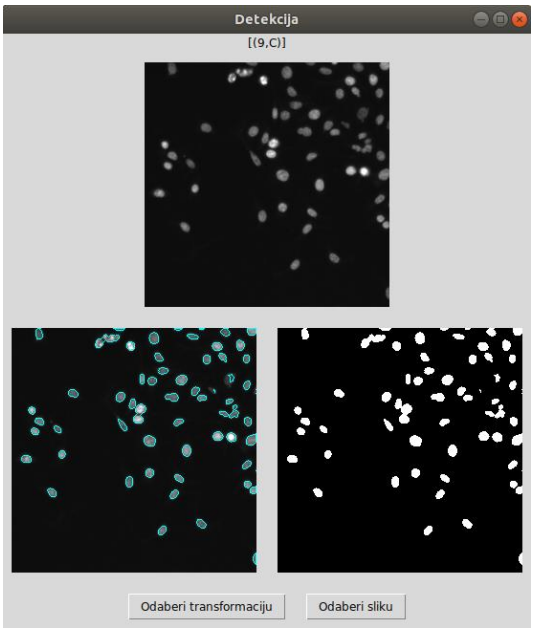
Za izradu ovog rada korišten je programski jezik Python. Razlog uporabe Pythona je njegova velika fleksibilnost i lagana prilagodba. Omogućuje laganu implementaciju željenih koncepata bez previše razmišljanja o samom oblikovanju i sintaksi izvornog koda.

Kako bi bio olakšan rad s algoritmima evolucijskog računanja, odabrana je Pythonova bibilioteka DEAP (*Distributed evolutionary algorithms in Python)* u kojoj postoje njihove implementacije.

Rad sa slikama prepušten je biblioteci OpenCV koja ima svoju implementaciju i raznim programskim jezicima pa tako i Pythonu.

Obje biblioteke o kojima ovisi programska podrška nude mogućnost optimiranog izvođenja. Prva to radi tako da izvođenje paralelizira unutar CPU-a, dok druga optimira vremenski zahtjevne operacije s matricama.

Za potrebe programske podrške izrađeno je jednostavno grafičko sučelje preko kojeg je moguće učitati transformacije te odabrati sliku koju je potrebno analizirati, tj segmentirati.



Slika 12 Grafičko sučelje za segmentaciju jezgara iz mikroskopskih slika

5. Rezultati

Prilikom testiranja programa, uzeti su neki predefinirani i neki promjenjivi parametri programa. Zadani u jednoj Python datoteci, moguće ih je promijeniti. Fiksni parametri genetskog algoritma i genetskog programiranja:

- Vjerojatnost mutacije jedinke = 0.5
- Vjerojatnost križanja = 0.7
- Maksimalna početna veličina jedinke = 5
- Vjerojatnost mutacije strukturnog elementa = 0.1

Svi ostali parametri bit će na neki način ispitani u ovisnosti o konačnoj dobroti rješenja. Određivanje dobrote radi se opisanom „jednostavnom tehnikom“ usporedbe piksela i broja pogođenih stanica. Razlog tomu je lakša interpretacija konačno dobivenog rješenja.

5.1. Usporedba slika bez i uz transformacije (GA)

Kako bi se provjerila ispravnost modela, potrebno je isti postupak izračuna dobrote primijeniti na slike na koje nisu primjenjivane morfološke transformacije i na one kojima su primjenjivane. Nad zadanom skupu od 30 nasumično iz baze preuzetih slika pokrenuto je evolucijsko pretraživanje najboljeg transformacijskog slijeda.

Zanimljiva je činjenica koja je proizišla iz analize rezultata. U inicijalnoj inačici programa, podržana je samo usporedba na temelju pogođenih piksela. Iako su mnoge od zadanih transformiranih slika dobile bolje pojedinačno rješenje naspram onih u kojima nije primjenjivana transformacija, srednja vrijednost tih rezultata zbog nekih izoliranih slučajeva težila je ka smanjenju, pa bi rezultati dobiveni tom metodom zapravo bili gori od najjednostavnijeg slučaja. To je ujedno očekivano i intuitivno. Komplicirano je pronaći model koji se dobro poklapa sa slikama različitih boja

i različitih dimenzija. Ovakvim pristupom model će se prilagoditi nekom određenom broju slika dok će za neke ipak raditi loše.

Stoga je podržana detekcija broja jezgara na slici, i primjer dobrote sljedova transformacija zadan je u nastavku (vrednovane slike razlikuju se od skupa za učenje):

	Bez transformacije ([])	[(C,SE_6), (O,SE_6), (D,SE_2)]	[(C,SE_{23}), (D,SE_2)]
Slika 1	63.6667	**83.8918**	82.6193
Slika 2	67.7129	**76.4651**	75.6988
Slika 3	79.1149	80.2424	**83.74439**
Slika 4	**82.6234**	78.5859	72.3139
Slika 5	**79.7997**	73.2030	68.5432
Slika 6	0.0000	**22.9637**	14.5365
Slika 7	3.9172	**79.5330**	62.4169

Tablica 2 Tablica rezultata segmentacije jezgara uz regularne strukturne elemente, prikaz je u postotcima

Vrijednost usporedbe zadana je kao postotak u tablici, dok su transformacije ispisane na vrhu. Primjenjivani su samo regularni strukturni elementi. Sljedeći pokazatelj bit će isto postavljena usporedba na neregularnim elementima, raspisanim kao kvadratna matrica u obliku vektora.

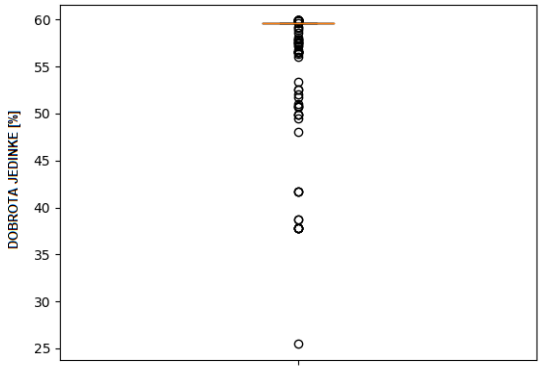
	Bez transformacije ([])	[(O,011 101 010)]	[(E, 01011 01110 00011 10111 00000), (D, 01011 00001 10010 00001 10101)]
Slika 1	63.6667	**89.5499**	80.1338
Slika 2	67.7129	88.1141	**89.4992**
Slika 3	79.1149	**86.1846**	84.7718
Slika 4	**82.6234**	70.7755	67.9225
Slika 5	79.7997	**81.1528**	75.5264
Slika 6	0.0000	0.0000	**20.5092**
Slika 7	3.9172	75.8351	**76.0885**

Tablica 3 Tablica dobrote segmentacije uz neregularne strukturne elemente

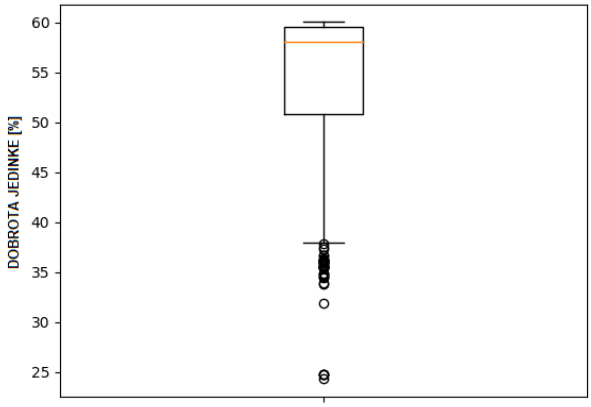
Iako je prikazano samo mali dio rezultata, već iz ovog primjera vidljivo je da transformacije ipak pomažu pri segmentaciji slike. Premda nije imala sve najbolje rezultate iz primjera, prosječno najbolja jedinka bila je ona posljednja s neregularnim strukturnim elementom veličine 5. Taj podatak je ponovno prirodan i vrlo intuitivan, genetskom algoritmu je dana veća sloboda i tu se tek vidi njegova moć i robusnost.

5.2. Kretanje jedinki tijekom evolucije

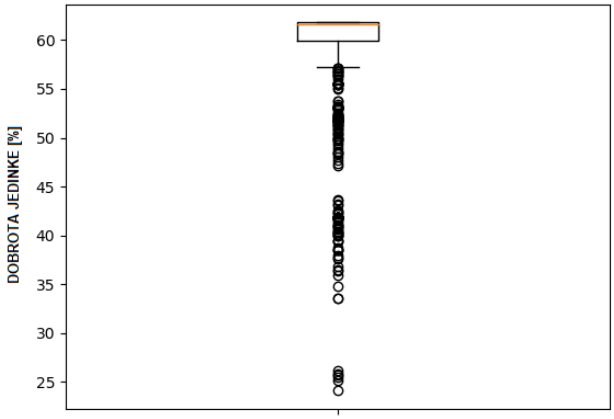
Vrijedan podatak je spoznaja kako se kroz generacije ponašala dobrota jedinki s različitom veličinom populacije. Taj podatak otkriva koliko brzo algoritam ulazi u lokalni optimumu, te koliko se od njega odmakne. Algoritam je pokretan kroz 600 evaluacija, vrednovan na 30 slika, s promjenjivim populacijama. Za strukturni element uzet je neregularni element veličine 5 zbog veće prilagodljivosti.



Slika 13 Prikaz raspodjele jedinki uz populaciju veličine 10

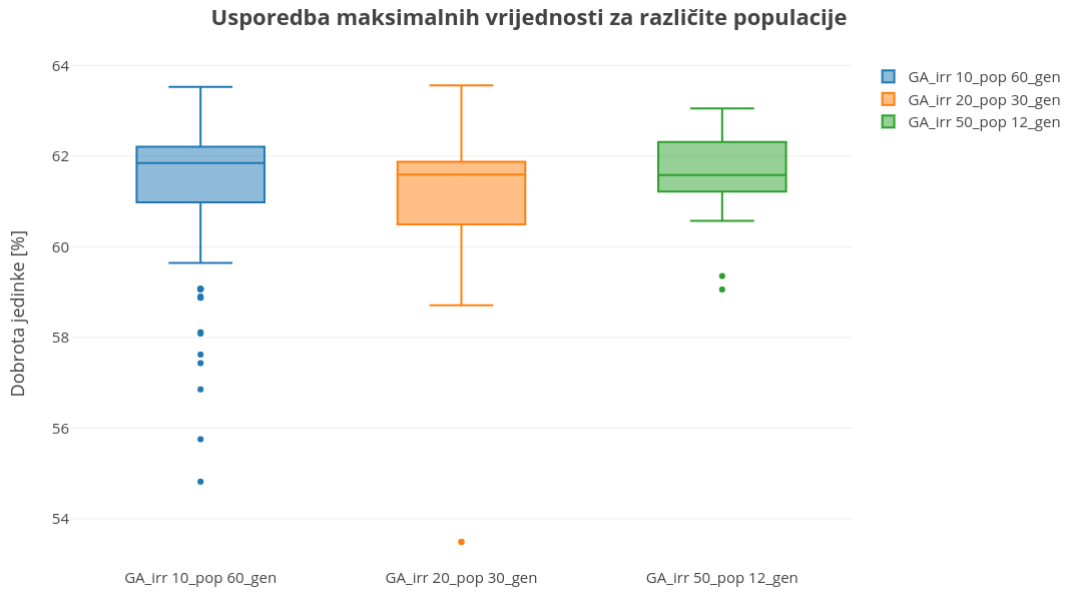


Slika 14 Prikaz raspodjele jedinki uz veličinu populacije 20



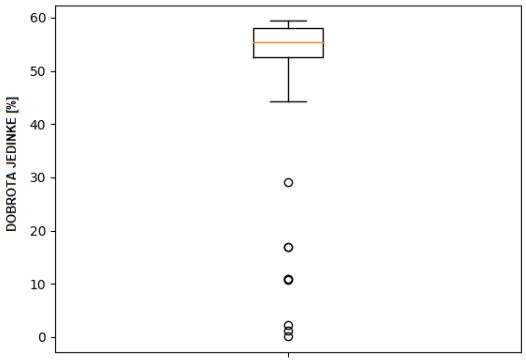
Slika 15 Raspodjela jedinki uz veličinu populacije 50

Kroz prethodne slike lako se dolazi do zaključka da jedinke stvorene generacijama populacije veće veličine imaju tendenciju pomaka ka boljim rješenjima, razlog tomu je bolja obuhvaćenost prostora pretraživanja. Ipak, Slika 16 dokazuje da u kretanju njihovih maksimalnih vrijednosti ne možemo izvući neki statistički značajan zaključak o kvaliteti najboljih rješenja u ovisnosti o veličini populacije.



Slika 16 Usporedba maksimalnih vrijednosti za različite populacije uz 600 evaluacija

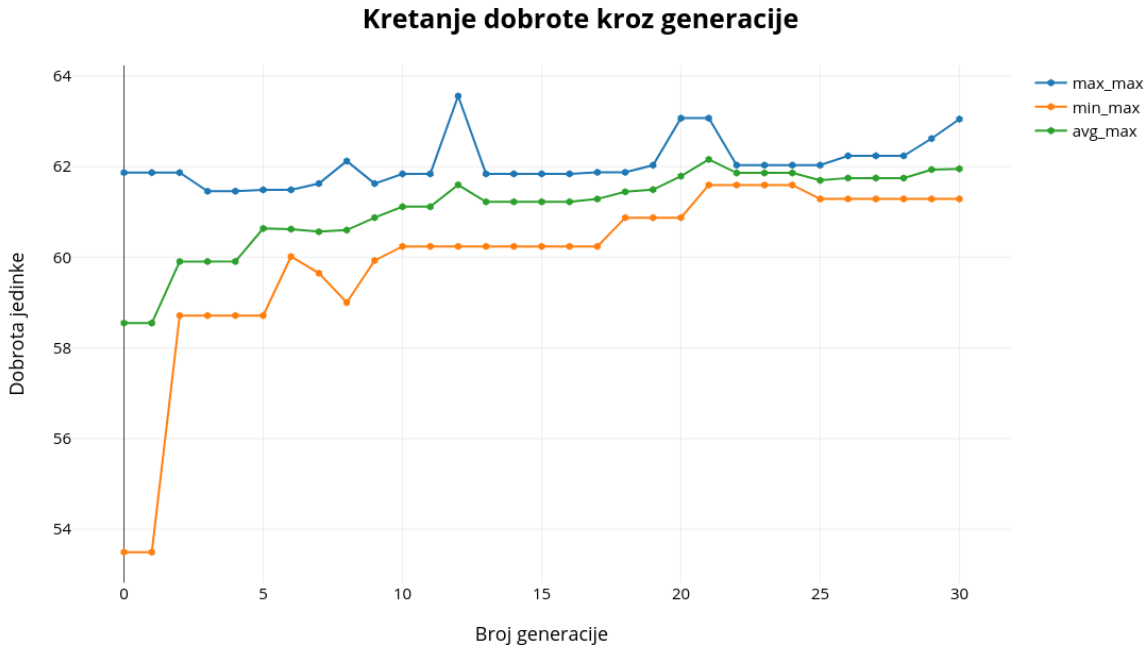
Dodatno, dobrota i veličina populacije neće biti u strogo povezanoj vezi. Algoritam je pokretan za veličinu populacije 200 jedinki i rješenja koja su pronađena nisu se pretjerano razlikovala od veličine 50, iako je bolje obuhvaćen prostor pretraživanja. Važan je trend koji postižu jedinke u maloj veličini populacije. One lakše upadaju u lokalni optimum i njihovo rješenje varira oko jedne točke, iako su mogući neki sporadični odmaci, dok se kod veće populacije u skupu održavaju raznovrsna rješenja.



Slika 17 Prikaz raspodjele jedinki u GP-u s populacijom veličine 10

U gornjoj slici predstavljen je podatak kretanja jedinki u genetskom programiranju. Zbog velike vremenske potrošnje potrebne da se obrade sve slike i izvrši stablo algoritam je pokrenut s veličinom populacije 10 kroz 20 generacija. Dobiveni rezultati približavaju se prethodno navedenim primjerima i uglavnom slijedi model genetskog algoritma unatoč prethodno opisanim ograničenjima.

Vrijedno je pogledati kako se kreće maksimalna vrijednost kroz generacije algoritma. Na Slici 18 grafički je prikazano kretanje jedinki maksimalne vrijednosti iz više pokretanja algoritma koji radi s 20 jedinki u populaciji i neregularnim strukturnim elementima.

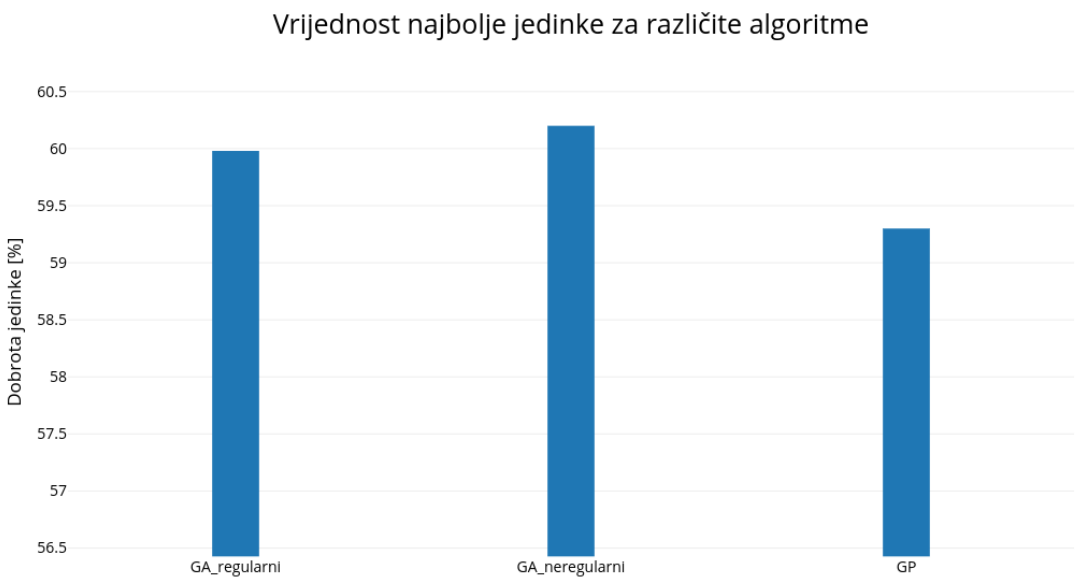


Slika 18 Kretanje maksimalne vrijednosti dobrote kroz generacije

Zbog jednostavnosti konačnog rješenja rijetko se dolazi do novih maksimalnih vrijednosti i ta vrijednost se često postiže u početnoj generaciji. Ipak, prikaz otkriva da bi uz veći broj evaluacija program eventualno pronašao i bolja rješenja te da uvjet zaustavljanja nije postignut. Testiranje takvih stvari onemogućeno je vremenskom ograničenosti i neelitističkim pristupom unutar biblioteke.

5.3. Kretanje i usporedba jedinki

Daljnja analiza uključuje pregled razlike rješenja koja su dobivena tijekom više pokretanja algoritama. Usporedba je načinjena nad genetskim algoritmom s regularnim i neregularnim elementima i sa stablom genetskog programiranja.

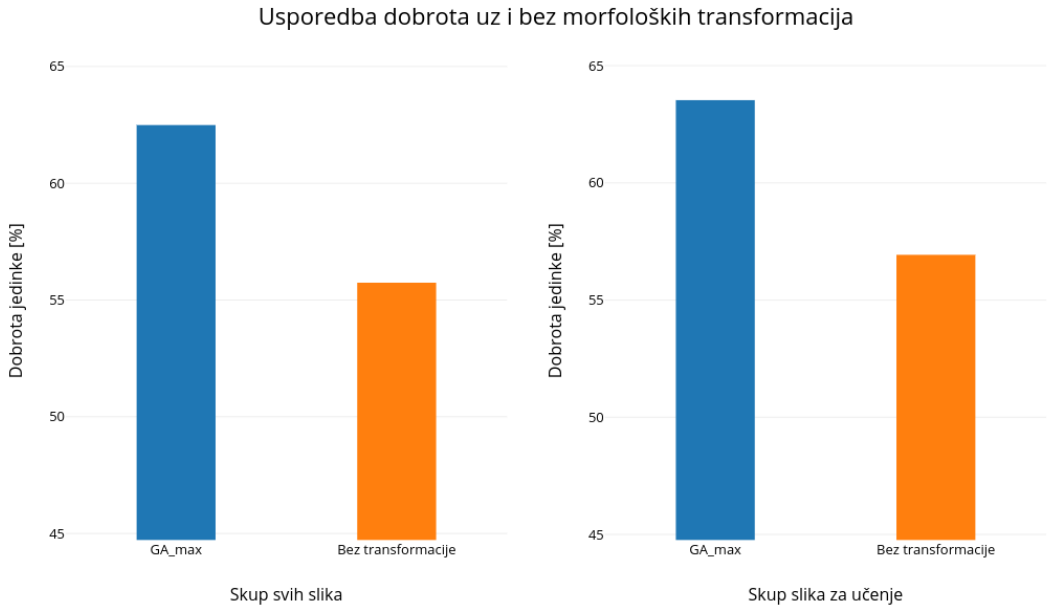


Slika 19 Odnos najboljih jedinki pri više pokretanja programa s predefiniranim parametrima na skupu za učenje od 30 slika

Dobivena rješenja ne razlikuju se pretjerano, ali to je uzrokovano činjenicom da je nad njima primijenjena ista taktika uz drugi algoritam pretrage. Također, udio u toj činjenici ima način evaluacije koji se oslanja na postotak pa razlike svodi na desetinke vrijednosti koje se čine zanemarivima, a u nekom realnom okruženju mogu igrati važnu ulogu.

Dodatan uzrok male razlike između rješenja je kratko vremensko izvođenje i velika konvergencija ka lokalnom optimumu.

Ovakvi rezultati ishod su evaluiranja 30 nasumično izabranih slika iz baze, za neke druge slike dobiveni rezultati izgledali bi potpuno drugačije (razlikovali bi se u vrijednosti funkcije dobrote).

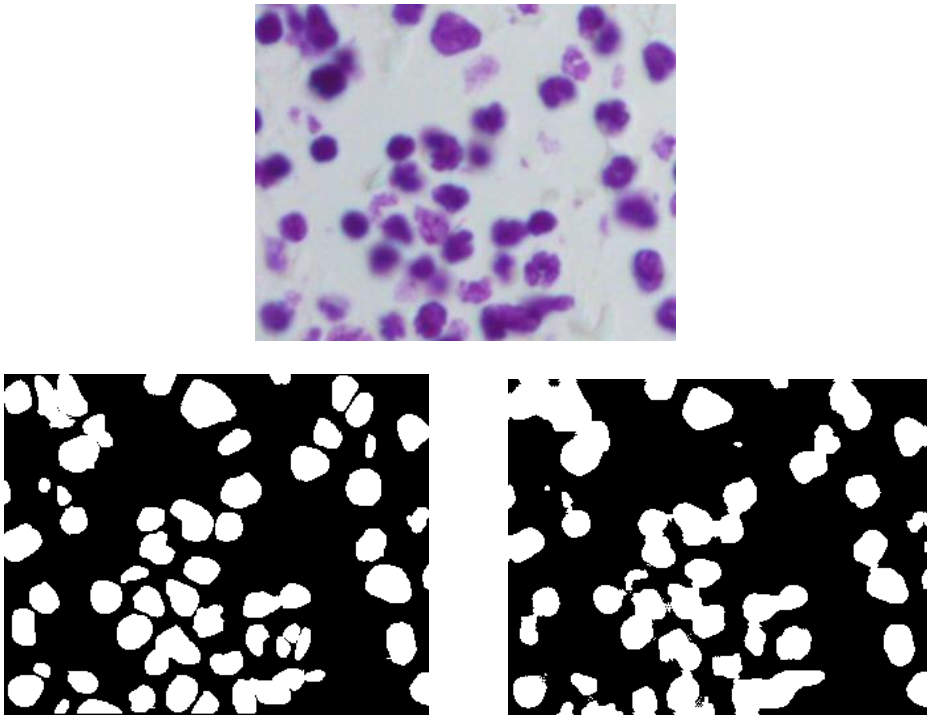


Slika 20 Prikaz odnosa prosječne dobrote slika uz primijenjene transformacije i slika bez transformacija, lijevi graf prikazuje ocjenu dobrote na svim slikama, a desni na slikama za učenje

Iako algoritam u normalnom okruženju sa slikama različitih boja i oblika ne postiže dobrotu veću od 62%, taj rezultat je ipak napredak ako se gleda relativna razlika s postupkom segmentacije koji se primjenjuje bez prethodne morfološke transformacije.

5.4. Konačna rješenja

Potvrda koncepta vidljiva je na pokretanju algoritma nad jednom slikom iz ulaznog skupa. Algoritam će se prenaučiti za tu sliku i neće biti u stanju izdvojiti željene segmente na ostalima, ali važna je to činjenica u dokazivanju rada modela.



Slika 21 Gore prikazana slika prije segmentacije, dolje lijevo zadana maska, dolje desno slika nakon transformacije i segmentacije

Jedinka koja je dobivena izgledala je sljedeće:

[(OPEN, 10101 000000 10100 00001 11001)]

Na jednoj slici vrijednost dobrote prije primjene transformacija bila je **0.43**, dok je nakon evolucije i određivanja prikladne transformacije vrijednost dobrote porasla na **0.58**.

Vidljivo je da je uspješno izdvojio neke od jezgara, i vidljiv je pokušaj razdvajanja stanica, kako bi se pravilno izdvojila preklapanja. Ipak ovom tehnikom dobiti taj rezultat bio bi preoptimističan ishod.

Pokretanjem algoritma nad većim skupom slika, dalo je drugačije rezultate. Ovaj puta neke slike dobile su bolju segmentaciju, dok su neke prošle gore zbog primijenjenih transformacija. Razlog tome krije se u činjenici da se na slici nalaze tri dominantne boje, zbog čega Otsuov postupak nailazi na probleme (histogram je trimodalni).

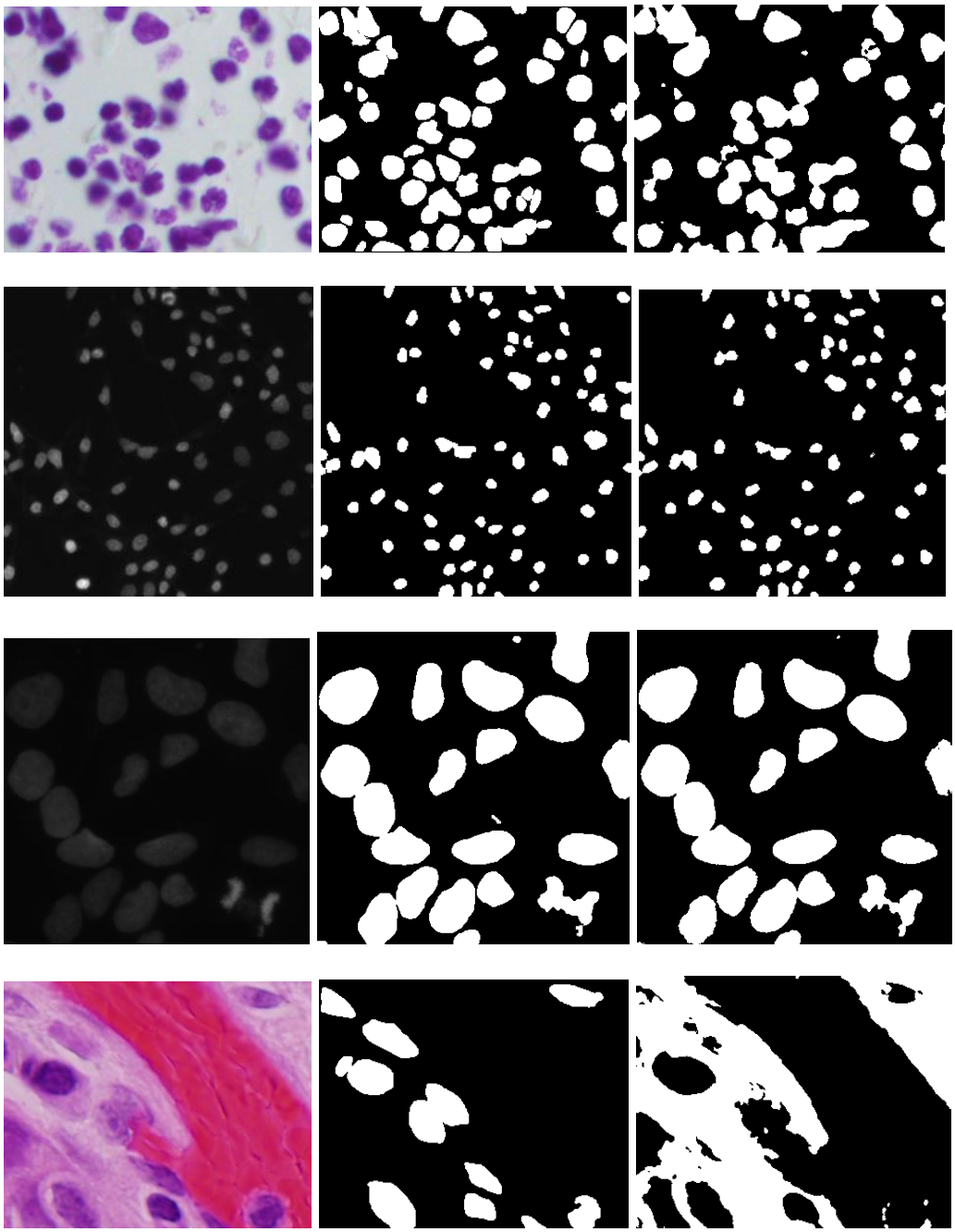
Konačno dobivena jedinka izgleda ovako:

[(E, 01011 01110 00011 10111 00000),

(D, 01011 00001 10010 00001 10101)]

Prosječna vrijednost dobrote slika prije no što je nad njima odrađena transformacija je **0.51**, dok je nakon primjene transformacija vrijednost prosječne dobrote porasla na **0.62**.

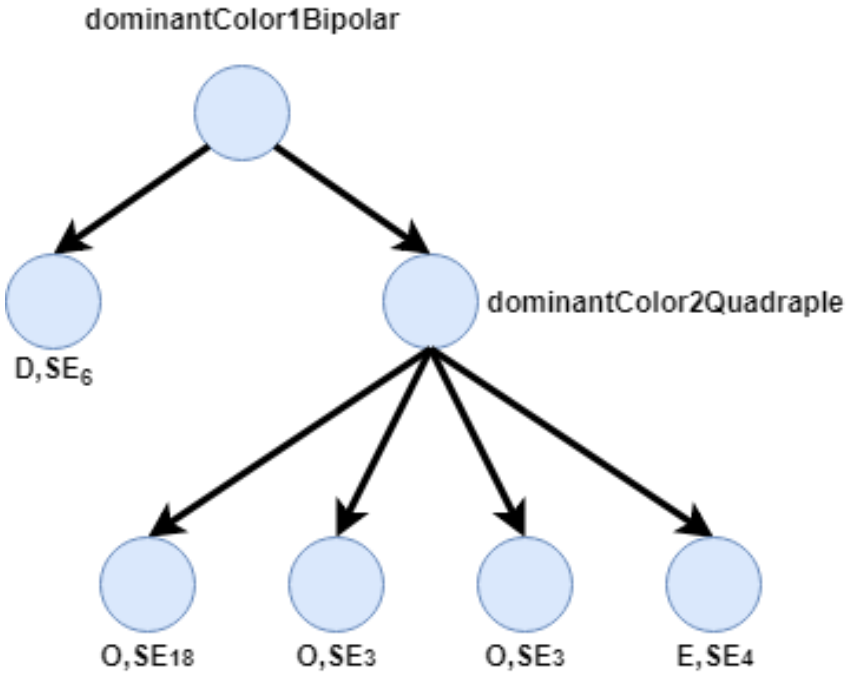
Slike koje su rezultat segmentacije zadane su u nastavku:



Slika 22 Izgled slika iz skupa za provjeru. Slike u kojima se miješaju tri dominantne boje lošije reagiraju na transformacije

Posljednje što je važno naglasiti su male veličine jedinki koje su produkt konačnog rješenja. Takvo što je uvjetovano dodavanjem dodatne transformacije pa se uvelike mijenja karakteristika neke od slika u skupu za učenje te se takva osjetljivost odražava na konačan rezultat.

Ta karakteristika olakšava posao stabla nastalog genetskim programiranjem, jer je ograničen na jednu transformaciju. Dodatno, genetsko programiranje bolje radi nad jedinkama s manjom razgranatosti zbog toga što su grane stabla ovisne o samo jednoj značajki slike (boji, intenzitetu) pa je nepotrebno više puta prolaziti kroz iste značajke.



Slika 23 Izgled najboljeg rješenja genetskog programiranja

6. Zaključak

Ideja za primjenom evolucijskog računanja pri segmentaciji biomedicinskih slika čini se razumnom zbog velike fleksibilnosti i modularnosti algoritma. Ipak, pored tehnologija koje su se pokazale kao standard u današnjoj analizi slika, ova tehnika postiže inferiornije rezultate.

Primjenjivani problem općenito je težak i znanstvenici diljem svijeta još uvijek ne postižu savršene rezultate. Zbog toga se cilj prebacio na usavršavanje tehnika koje će omogućiti pravilnu segmentaciju slika pojedinih kategorija i sličnih značajki.

Model prikazan u ovom radu iako nije uspio usavršiti tu tehniku, nad manjim skupom slika prikazan je ispravnim i gledajući globalnu sliku bolji je od onog bez pretprocesiranja slike za analizu. Algoritam je pokretan nad skupom za učenje i nakon obrade rezultati su uspoređivani sa skupom za testiranje. Evaluacija dobrote nad slikama nakon primjene transformacija uvijek je bila bolja od primjene odsijecanja praga bez transformacija.

Budući rad nad modelom i algoritmima uključio bi paralelizaciju izvođenja i izvođenje u programskom okruženju koje ne pati od sporog izvođenja. Takav pristup omogućio bi češće i brže testiranje parametara. Također, budući rad bio bi okrenut prema izradi vlastitih biblioteka evolucijskog računanja zbog ograničenosti postojećih na vlastito zamišljene implementacije.

Zadnje što je važno istaknuti jest predanost radu u biomedicinske svrhe. Teme poput ove potrebno je duboko ukorijeniti u znanstvene radove na području računarske znanosti. Razvoj tehnologije može uvelike utjecati na živote milijuna ljudi.

7. Literatura

[1]. Wikipedia: „Enroom's law“, https://en.wikipedia.org/wiki/Eroom%27s_law7

[2]. D.B. Fogel: „Nils Barricelli - artificial life, coevolution, self-adaptation“, veljača 2006.

[3]. Shaho Shahbazpanahi: “*Learning Image Enhancement and Object Localization Using Evolutionary Algorithms*”, University of Ontario Institute of Technology, ožujak 2014

[4]. Kaggle 2018. Science Bowl: https://www.kaggle.com/c/data-science-bowl-2018

[5]. Olaf Ronneberger, Philipp Fischer, and Thomas Brox: “ U-Net: Convolutional Networks for Biomedical Image Segmentation”, University of Freiburg

[6]. Marko Čupić: “ Prirodom inspirirani optimizacijski algoritmi. Metaheuristike. ”, Sveučilište u Zagrebu, kolovoz 2013.

[7]. Alaa Sheta, Malik S. Braik, Sultan Aljahdali: “ AljahdaliGenetic Algorithms: A Tool for Image Segmentation ”

[8]. Ming Yu: “ Image segmentation using genetic algorithm and morphological operations ” *,Iowa State University*

[9]. Angela Ribeiro, Juan Ranz, Xavier P. Burgos-Artizzu, Gonzalo Pajares ,Maria J. Sanchez del Arco, Luis Navarrete:“ An Image Segmentation Based on a Genetic Algorithm for Determining Soil Coverage by Crop Residues ”

[10]. Y.L. Varol: “ Using genetic algorithms for model-based object recognition ” , University of Nevada, Reno

[11]. Wikipedia: “Evolutionary Computing” , https://en.wikipedia.org/wiki/Evolutionary_computation .

8. Sažetak

Tema ovog rada je problem detekcije jezgara stanica iz mikroskopskih slika pristupom temeljenim na metodama morfoloških transformacija optimiziranih evolucijskim algoritmima i metodom odsijecanja praga. Problem se sveo na optimiranje pravilnog slijeda morfoloških transformacija i njima pripadnog strukturnog elementa korištenog u obradi slike. Ostvarena je mogućnost evoluiranja tog slijeda koristeći genetski algoritam i genetsko programiranje, čije su značajke opisane i uspoređivane. Dobiveni rezultati pokazali su kako model nije dovoljno prilagodljiv da bi nadmašio do sada korištene tehnike segmentacije biomedicinskih slika, premda se pokazao kao rješenje koje na okruženju koje nije dodatno zakomplicirano daje brze i točne rezultate. Takvi konačni rezultati jednim dijelom učinak su spore obrade velike količine slika i vremenske ograničenosti na dodatna istraživanja.

Ključne riječi: Genetski algoritam, Genetsko programiranje, Detekcija jezgri stanica, Biomedicina, Segmentacija slike, Morfološke transformacije

9. Abstract

Main problem elaborated in this paper is detecting cell's nucleus within given microscopic photo using methods of morphology transformations optimized by evolutionary computing algorithms and thresholding methods for segmentation featuring elements. Optimization problem is finding the most suitable configuration of given morphology transformations and their given structuring element used in image segmentation. Realized versions of optimization techniques are genetic algorithm and genetic programming which features are compared in paper. Final results suggested that resulted model wasn't as successful as modern day used segmentation techniques. Although, in much simpler environment model seems to be suited for the task given. Usage of great number of images in segmentation is very time consuming and it was a barrier for delivering better results on this problem.

Key words: Genetic algorithm, Genetic programming, Nucleus detection, Biomedicine, Image segmentation, Morphology transformations