SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 953

EVOLUCIJA PRIKAZA SLIKE TEMELJENOG NA GEOMETRIJSKIM ELEMENTIMA

Nenad Krpan

Zagreb, lipanj 2009.

Sadržaj

1. Uvod

Neki problemi imaju jako veliko područje pretraživanja i bez naprednih algoritama ne bi ih bilo moguće riješiti u razumnom vremenu. Uloga genetskih algoritama je rješavanje takvih problema. Moguća rješenja kodiraju se u kromosome te, koristeći iste principe kao biološka evolucija, genetski algoritmi „evoluiraju“ rješenje. Konačna rješenja često nisu optimalna, nego je cilj dobiti dovoljno dobro rješenje.

Jedan od takvih problema je pretvaranje rasterske slike u sliku načinjenu od poluprozirnih poligona. U ovom radu opisano je kako je slika kodirana u kromosomu, tj. njen genotip i kako se iz kromosoma dobiva slika, tj. njen fenotip. Opisana je i programska implementacija takvog genetskog algoritma te razni genetski operatori korišteni za dobivanje slike. Obavljena su mjerenja za svakoga od njih kako bi se utvrdila učinkovitost.

Osim korištenja svih genetskih operatora odjednom, mjerena je i učinkovitost korištenja samo operatora mutacije.

2. Genetski algoritam

Genetske algoritme predložio je John Holland 1960-ih i razvio ih je sa svojim studentima i kolegama na Sveučilištu u Michiganu 1960-ih i 1970-ih. Cilj mu je bio formalno istražiti adaptacije koja se događaju u prirodi i pronaći način da ih preslika u računalni oblik.

Najkorisniji su za probleme kod kojih:

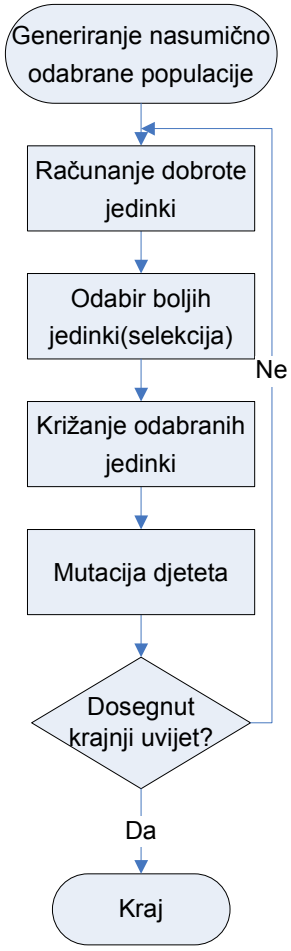
- je područje pretraživanja preveliko ili presloženo,
- nema dovoljno stručnog znanja iz domene problema, ili je to znanje teško iskoristiti za smanjenje područja pretraživanja,
- ne postoji matematička analiza problema,
- pokušaji s klasičnim metodama ne daju dobre rezultate.

Dobivena rješenja često ne mogu biti najbolja, već se nastoji dobiti rješenje koje je dovoljno dobro.

Važan dio programiranja genetskog algoritma je kodiranje mogućih rješenja u kromosome. Najčešće se kodiraju u nizove bitova jednake duljine, tako da se mogu jednostavnije križati i mutirati. Neke druge mogućnosti su nizovi promjenjive duljine i stablaste strukture.

Genetski algoritam počinje s populacijom kromosoma (npr. niz bitova), te računalnim ekvivalentima prirodnog odabira (engl. *natural selection*), križanja (engl. *crossover*) i mutacije stvara se nova populacija. Proces se ponavlja dok se ne ostvari neki krajnji uvjet (npr. vrijeme, broj generacija, dovoljno dobro rješenje).

Redoslijed operacija može se opisati dijagramom toka (Slika 2.1).



Slika 2.1 Diagram toka genetskog algoritma

Dobrota (engl. *fitness*) jedinke računa se heurističkom funkcijom i ona opisuje koliko je jedinka blizu cilju.

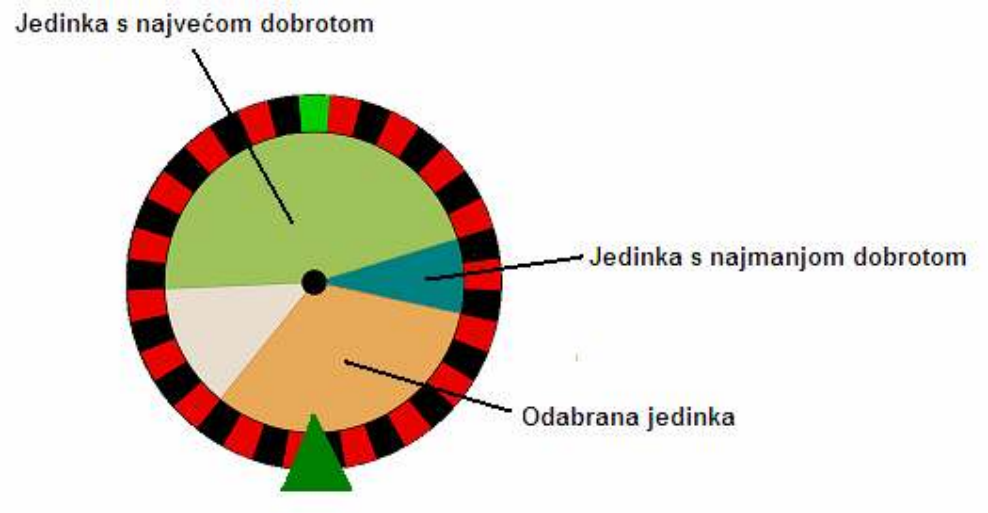
Svi genetski algoritmi imaju barem tri genetska operatora: selekciju, križanje i mutaciju.

2.1. Selekcija

Selekcija je biranje jedinki koje će graditi sljedeću generaciju i može se izvršiti na više načina. Cilj selekcije je istaknuti bolje jedinke, u nadi da će njihovo potomstvo imati još veću dobrotu. Selekcija treba biti uravnotežena; ako je prestroga (najbolji imaju jako veliku vjerojatnost da budu odabrani), neoptimalne jedinke s velikom dobrotom će preuzeti cijelu populaciju i smanjiti raznolikost potrebnu za daljnji napredak populacije; ako nije dovoljno stroga, populacija će presporo evoluirati.

Neke od vrsta selekcije su selekcija proporcionalna dobroti, turnirska, i selekcija rangiranjem.

Selekcija proporcionalna dobroti (engl. *fitness-proportionate selection*) se zove tako zato što je vjerojatnost da neka jedinka bude odabrana proporcionalna njenoj dobroti. Najčešća implementacija te metode je *roulette-wheel* gdje svaka jedinka dobiva dio na kolu ruleta koji je proporcionalan njenoj dobroti (Slika 2.2), zatim zavrtimo kolo onoliko puta koliko roditelja želimo dobiti. Nedostatak te selekcije je u tome što je na početku razlika u dobroti između jedinki velika, pa se dobre jedinke i njihova djeca proširuju po cijeloj populaciji te time ograničuju pretraživanje prostora rješenja na svoju okolinu (tzv. prerana konvergencija). Nakon nekog vremena razlika u dobroti se smanjuje i selekcija prestaje biti djelotvorna i evolucija se usporava.



Slika 2.2 Roulette-wheel selection

Algoritam se može implementirati na ovaj način:

1. zbraja se dobrota cijele populacije (*suma*)
2. odabire se nasumičan broj između 0 i *suma* (*br*)
3. prolazi se po svim jedinkama i zbrajaju se njihove dobrote, ako je trenutni zbroj veći od *br*, odabire se trenutna jedinka

Kako bi se povećali omjeri dobrota između jedinki, prije prvog koraka svim dobrotama može se oduzeti dobrota najlošije jedinke.

Turnirska selekcija radi tako da nasumično odabire nekoliko jedinki iz populacije i oni se natječu u „turniru“, gdje se na neki način odaberu bolje jedinke. Za razliku od druge dvije selekcije, turnirska selekcija ne mora prolaziti po svim jedinkama, te je zbog toga brža, iako u većini slučajeva računanje dobrote traje puno duže, tako da to nije toliko bitno. Korištenjem samo dijela populacije odjednom pokušava se spriječiti prebrza konvergencija.

Neki od mogućih načina odabira:

a) iz populacije se nasumično odabiru 2 jedinke, odabere se slučajan broj *r* od 0 do 1. Ako je $r<k$ (*k* – proizvoljan parametar za selekciju, npr. 0.4), onda se odabire bolja jedinka, inače se odabire lošija.

b) iz populacije se nasumično odabiru N jedinki, dvije najbolje se odabiru za roditelje, a najgora jedinka zamijeni se njihovim djetetom.

Selekcija rangiranjem (engl. *rank selection*) je slična turnirskoj selekciji po tome što vjerojatnost da neka jedinka bude odabrana jedino ovisi o tome da li je jedinka bolja ili lošija od drugih jedinki, ali ne i za koliko. Jedinke se sortiraju po dobroti, i jedino je bitno na kojoj poziciji se jedinka nalazi. Takvo zanemarivanje apsolutne vrijednosti dobrote je dobro jer sprječava probleme s konvergencijom, ali nekada može biti bitno znati da je neka jedinka daleko bolja od susjedne jedinke.

Pseudokod jedne verzije selekcije rangiranjem:

```
sortiranje_po_dobroti(populacija);
i=0;          //k=parametar selekcije = sansa da
dok(rand()>k) //                ce prvi biti odabran
{             //rand() => [0,1]
   i++;
   ako(i >= velicina_populacije) i=0;
}
index_odabrane_jedinke=i;
```

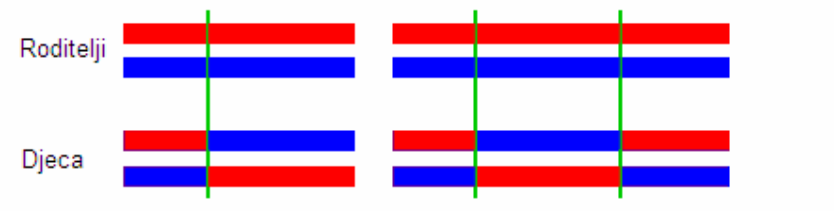
Samo zato što najbolje jedinke imaju najveću vjerojatnost da budu odabrane ne znači da će i biti odabrane. Može se dogoditi da se najbolje rješenje izgubi, zato se uvodi **elitizam**. Elitizam se ostvaruje tako da se najbolja jedinka (ili više njih) nepromijenjeno prenese u sljedeću generaciju. Mnogi istraživači su ustanovili da elitizam znatno poboljšava učinkovitost genetskog algoritma. Neke selekcije mogu u sebi implicitno sadržavati elitizam (npr. prethodno opisana turnirska selekcija pod b).

2.2. Križanje

Križanje (*crossover*) je kombiniranje gena dviju jedinki (roditelja), kako bi se stvorila treća (dijete). Način križanja ovisi o vrsti kodiranja kromosoma.

Križanje kromosoma predstavljenih nizovima

Križanje s jednom točkom prekida (Slika 2.3) se ostvaruje tako da se odabere jedna točka na kromosomu i dijete se stvara od početka kromosoma jednog roditelja, i kraja kromosoma drugog roditelja. Nedostatak kod takvog križanja je u tom što segment koji se mijenja uvijek sadrži krajnju točku, i geni koji nisi korisni se prenose zajedno s korisnima. Kako bi se to spriječilo koristi se križanje s dvije točke prekida (Slika 2.3).

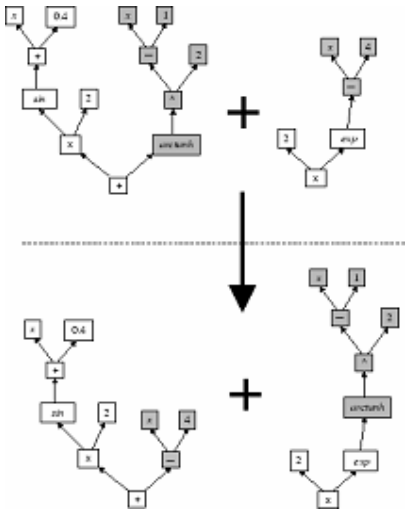


Slika 2.3 Križanje s jednom točkom prekida (lijevo) i križanje s dvije točke prekida (desno)

Jednoliko križanje (engl. *uniform crossover*) uzima bit po bit iz roditelja, i za svaki bit postoji 0.5 vjerojatnosti da će biti iz jednog roditelja i 0.5 iz drugog.

Križanje kromosoma predstavljenih stablom

Stabla se najčešće koriste za kodiranje kromosoma u genetskom programiranju. Na svakom roditelju odabire se nasumično jedan čvor u stablu, i od jednog roditelja se uzima cijelo stablo bez toga čvora, a od drugog se uzima samo grana čiji je taj čvor početak (Slika 2.4.)



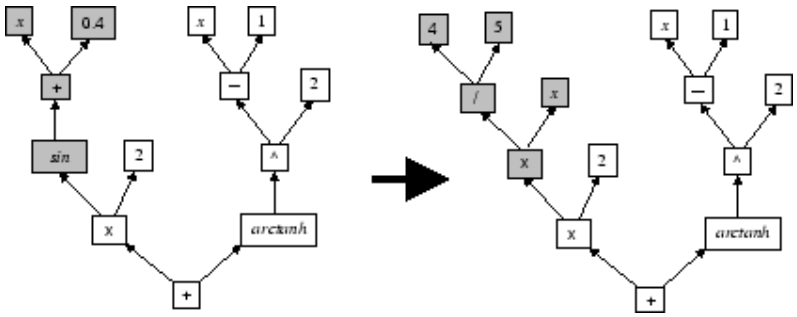
Slika 2.4 Križanje stabla

2.3. Mutacija

Nakon što se križanjem napravi nova jedinka, na nju se primjeni mutacija, kako bi se uvela raznolikost u populaciju. Kao i kod križanja, mutacija ovisi o načinu kodiranja kromosoma. Obično se pri pokretanju genetskog algoritma zadaje postotak mutacije *m* (npr. 0.005).

Ako imamo kromosom predstavljen nizom, onda za svaki bit iz kromosoma postoji vjerojatnost *m* da će se preokrenuti (0→1, 1→0).

Ako je kromosom predstavljen stablom, onda ne mutiramo jedan bit, nego nasumično odaberemo jedan čvor i s vjerojatnošću *m* ga mutiramo tako da izbrišemo čvor i sve grane koje izlaze iz njega, i napravimo na tom mjestu novu granu (Slika 2.2.5).



Slika 2.2.5 Mutiranje stabla

3. Prikaz slike poligonima

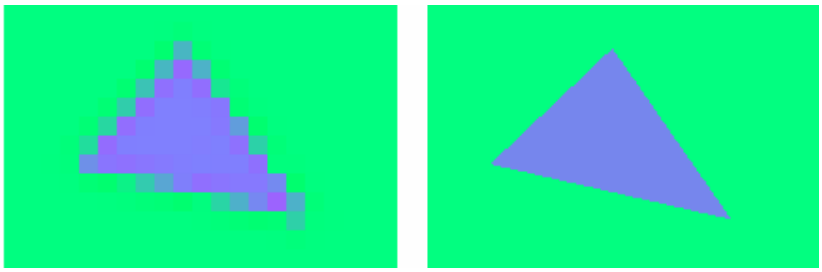
Cilj ovog rada je pretvoriti sliku u rasterskom obliku (napravljenu od piksela), u vektorski oblik. Slika u vektorskom obliku bila bi napravljena od poligona koji bi bili definirani točkama, bojom (RGB) i prozirnošću. Boja pozadine slike također je promjenjiva.

Na slici 3.1 vidimo primjer jedne rasterske slike i vektorske slike napravljene od nje, koja se sastoji od 60 poligona od kojih svaki ima 6 točaka.

Slika 3.1 Rasterska slika (lijevo) i slika napravljena od poligona (desno)

Poligoni su prozirni kako bi se moglo dobiti više nijansi boja nego što ima poligona. Prednost vektorskih slika je ta što se mogu skalirati bez gubitka kvalitete.

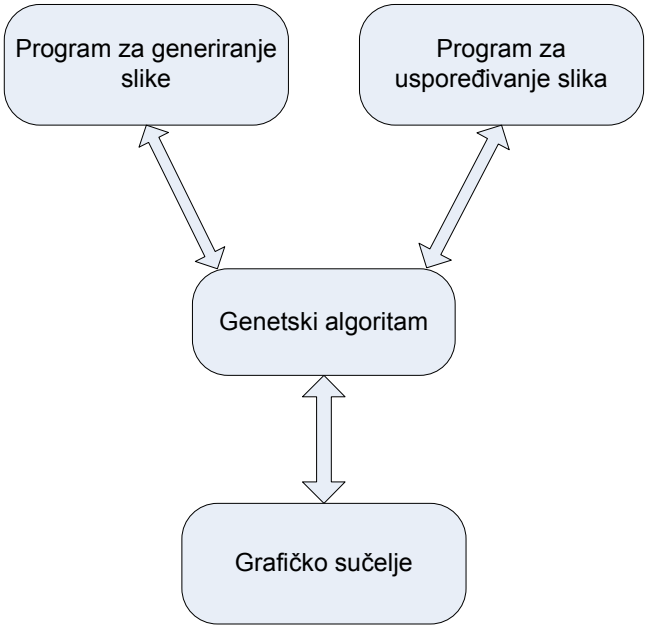
Nisko rezolucijske slike se mogu i popraviti vektorizacijom (Slika 3.2)



Slika 3.2 Rasterska slika (lijevo) i slika dobivena vektorizacijom (desno)

4. Opis programske implementacije

Implementacija genetskog algoritma za pretvaranje rasterske slike u sliku složenu od poluprozirnih poligona napravljena je od više manjih programa. Programi su zajedno spojeni u jednu cjelinu (Slika 4.1).



Slika 4.1 Programi

Genetski algoritam se može pokretati izravno iz komandne linije ili pomoću grafičkog sučelja. Programi za generiranje i uspoređivanje slika koriste se unutar genetskog algoritma. Postoji više inačica tih programa i pri pokretanju se može odabrati ona koja će se koristiti.

4.1. Generiranje slike

Program za generiranje slike služi za to da genotip jedinke pretvara u fenotip. Koristi se izravno iz genetskog algoritma, ali se može pokretati i iz komandne linije kako bi se iz *.poly* datoteka, napravljenih za vrijeme rada genetskog algoritma, stvarala *.bmp* slika.

Napravljene su dvije verzije programa, s OpenGL-om, i sa CImg bibliotekom. Verzija sa CImg-om jednostavnija je za implementirati te, budući da koristi samo standardni C++, radit će svugdje, ali je zato verzija s OpenGL-om puno brža. Iz tablice 4.1 vidimo da je OpenGL brži i da radi jednako brzo za različite brojeve poligona, dok se CImg znatno uspori pri crtanju više poligona i točaka.

broj poligona/točaka	OpenGL	CImg
50 / 6	78 ms	236 ms
100 / 6	79 ms	370 ms
100 / 10	79 ms	648 ms

Tablica 4.1 Mjerenja generiranja slike od 334x520 piksela

Program se može pokretati na više načina:

```
ime_programa.exe file1.poly file2.bmp
```

Poligoni se učitavaju iz *file1.poly* i izrađuje se slika *file2.bmp.*

```
ime_programa.exe file2.bmp
```

Poligoni se učitavaju sa standardnog ulaza (*stdin*) i izrađuje se slika *file2.bmp*.

```
ime_programa.exe
```

Poligoni se učitavaju sa standardnog ulaza i slika se šalje na standardni izlaz (*stdout*).

```
ime_programa.exe --server-mode
```

Radi jednako kao i bez parametara, ali u petlji bez gašenja. Genetski algoritam će ga koristiti na taj način zato što se najviše vremena troši na pokretanje programa.

ime_programa.exe može biti *Render_image_CImg.exe* ili *render_image_OpenGL.exe.*

Ulazni podaci (i *.poly* datoteke) su rezolucija slike, boja pozadine, točke poligona i boje i prozirnosti poligona, u binarnom obliku bez predznaka. Točan redoslijed može se vidjeti u tablici 4.2.

podatak	rezolucija slike		boja pozadine			broj poligona	1. poligon										…
	x	y	R	G	B		broj točaka	R	G	B	A	x_1	y_1	…	x_n	y_n	
veličina/bajt	2	2	1	1	1	2	1	1	1	1	1	2	2		2	2	

Tablica 4.2 Raspored bajtova u *.poly* datoteci

Iz toga se vidi da najveća rezolucija može biti 65535x65535, najviše može biti 65535 poligona na slici, a svaki poligon može imati najviše 255 točaka.

Koordinate točaka mogu biti [0,65535] i to se skalira na [0,rezolucija] kako bi se lakše moglo križati i mutirati kromosome i kako bi se mogla promijeniti rezolucija slike bez dodatnog skaliranja točaka.

4.2. Uspoređivanje slika

Program za uspoređivanje slika prima dvije slike u *bmp* formatu i na standardni izlaz ispisuje njihovu razliku. Razlika se računa kao srednja kvadratna pogreška (4.1). Moguće vrijednosti su [0,65025].

$$D = \frac{1}{3n} \sum_{n} \left(\Delta R^2 + \Delta G^2 + \Delta B^2 \right) \tag{4.1}$$

Program se koristi za računanje dobrote jedinki na taj način da se fenotip jedinke uspoređuje s ciljnom slikom.

Program se može pokretati na više načina:

```
Compare_BMP_RGB_MSE.exe file1.bmp file2.bmp
```

Učitava slike *file1.bmp* i *file2.bmp* i vraća njihovu razliku na standardni izlaz.

```
Compare_BMP_RGB_MSE.exe file1.bmp
```

Učitava sliku *file1.bmp,* a drugu sa standardnog ulaza i vraća njihovu razliku.

```
Compare_BMP_RGB_MSE.exe file1.bmp --server-mode
```

U *file1.bmp* je ciljna slika, a na standardni ulaz šaljemo slike (fenotipove) za uspoređivanje s njom.

4.3. Genetski algoritam

Ova komponenta je glavni program koji pokušava evoluirati sliku što bližu zadanoj koristeći samo poligone. Unutar funkcije za dobrotu koristi prethodna dva programa: prvo šalje genotip jedinke programu za generiranje slike te primi natrag fenotip, zatim fenotip šalje programu za uspoređivanje koji vraća razliku između njega i zadane slike – dobrotu. Traži se minimum.

Pri pokretanju programa mogu se zadati razni parametri: veličina populacije, postotak mutacije, broj poligona na slici, broj točaka po poligonu, više operatora za selekciju, križanje i mutaciju i parametri za njih te uvjeti zaustavljanja. Za popis parametara program se pokreće sa zastavicom `--help`.

Kada program krene, stvara se novi direktorij koji u imenu ima ime slike i trenutni datum i vrijeme. U njega se kopira zadana slika i stvara se *.log* datoteka u koju se upisuju zadani parametri te se svake generacije upisuje dobrota najbolje jedinke i prosječna dobrota populacije. U istom direktoriju svake generacije u kojoj ima poboljšanja stavlja se jedna *.bmp* slika najbolje jedinke, i *.poly* datoteka s poligonima.

Svake iteracije selekcija odabire dvije jedinke koje će biti roditelji i jednu jedinku koja će se zamijeniti. Sa funkcijom za križanje spajaju se kromosomi roditelja te se dobivenom jedinkom zamjenjuje odabrana jedinka. Na dijete se zatim primjenjuje operator mutacije i izračunava se dobrota.

Operatori se biraju tako da se navedu opcije *--selection=<int>*, *--crossover-operator=<int>* i *--mutation-operator=<int>* na broj željene funkcije, koje možemo vidjeti ako pokrenemo program s opcijom *--print-operators.*

Selekcija

Ostvareno je nekoliko selekcija:

1. *tournament* - opisana u poglavlju 2.1 - turnirska selekcija pod b), može se birati broj natjecatelja u turniru.
2. *roulette wheel* - opisana u poglavlju 2.1.
3. *rank* - opisana u poglavlju 2.1, može se birati vjerojatnost da će prvi biti odabran.

Prve dvije selekcije implicitno u sebi imaju elitizam.

Križanje

Ostvareno je nekoliko operatora križanja:

1. *polygonwise* - dijete se dobiva tako da se iz oba roditelja uzima nekoliko nasumično odabranih poligona, a boja pozadine je jednoliko križanje roditeljevih boja.
2. *pointwise* - izrađuje se *p* poligona u koje se stavlja nekoliko nasumično odabranih točaka iz roditelja, boja novog poligona je jednaka boji poligona iz kojeg je jedna točka novog poligona, a boja pozadine je jednoliko križanje roditeljevih boja.
3. *match random polygon* - za svaki poligon iz prvog roditelja, nasumično se odabire poligon iz drugog roditelja i kombinacijom njih napravi se poligon za dijete. Uzima se nekoliko točaka iz oba roditelja, a boja je jednoliko križanje boja poligona. Boja pozadine je jednoliko križanje roditeljevih boja.
4. *match closest poligon* - jednako kao i za *match random polygon*, osim što se iz drugog roditelja ne odabire nasumičan poligon, nego onaj kojem je težište najbliže težištu poligona iz prvog roditelja.

Mutacija

Ostvareno je nekoliko operatora mutacije:

1. *polygonwise* - za svaki poligon postoji vjerojatnost *m* da će biti zamijenjen novim slučajnim poligonom. Boja pozadine se mutira[1].
2. *pointwise simple* - za svaku točku na slici postoji vjerojatnost *m* da će se zamijeniti novom točkom i boja poligona se mutira. Boja pozadine se mutira.
3. *pointwise advanced* - za svaki poligon postoji vjerojatnost *m* da će biti mutiran. Mutira se na taj način da mu se dodaje ili briše točka ili zamjenjuje neka postojeća točka novom točkom i mutira mu se boja. Boja pozadine se mutira.

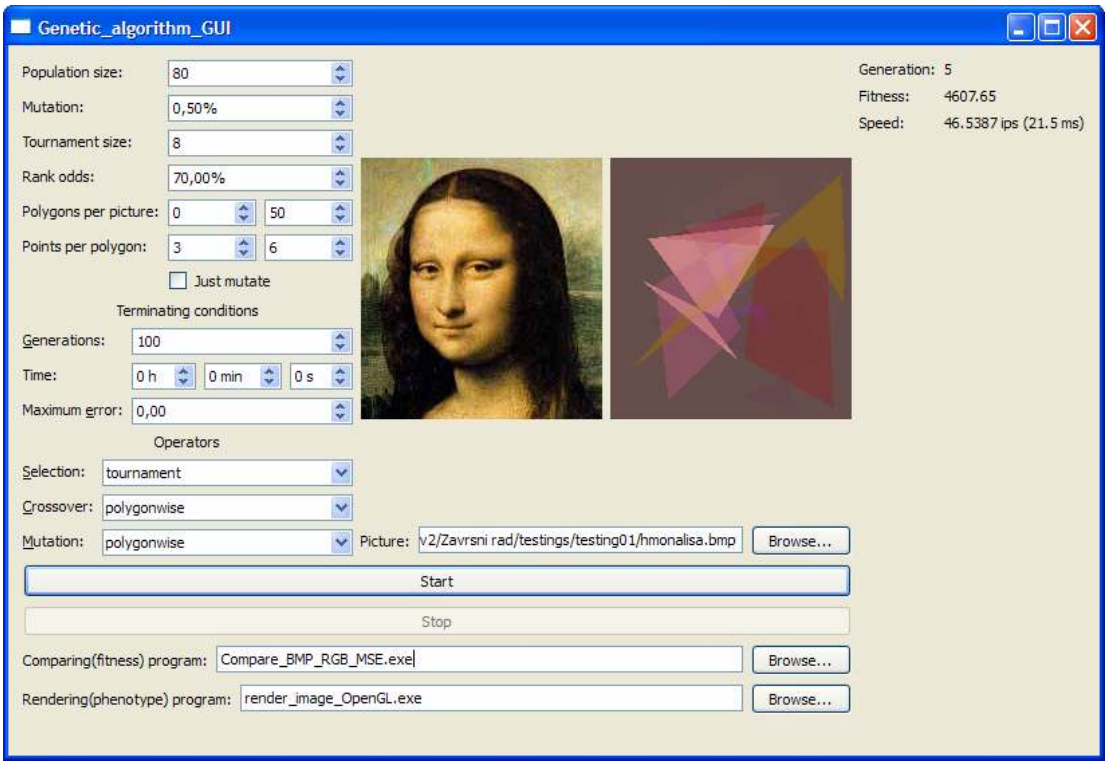
[1] Sve boje se mutiraju tako da za svaki bit boje postoji vjerojatnost *m* da će se preokrenuti.

4. *point displacement* - boja pozadine i boje poligona se mutiraju. Za svaku točku postoji vjerojatnost *m* da će se mutirati. Mutirane točke se pomiču sa svoje trenutne pozicije u smjeru φ, za radijus *r*. φ je nasumično odabran broj s jednolikom razdiobom od 0° do 360°. *r* je nasumično odabran broj s razdiobom 0.2ln(x) i vjerojatnost da će biti manji od 20% dijagonale slike je oko 70%.

Postoji još i opcija *--just-mutate*, gdje se ne koristi križanje, nego samo mutacija. Koriste se samo dvije jedinke, jedna se mutira i bolja se presnimi preko lošije.

4.4. Grafičko sučelje

Kako bi bilo lakše pokretati genetski algoritam, napravljeno je grafičko sučelje (Slika 4.2).

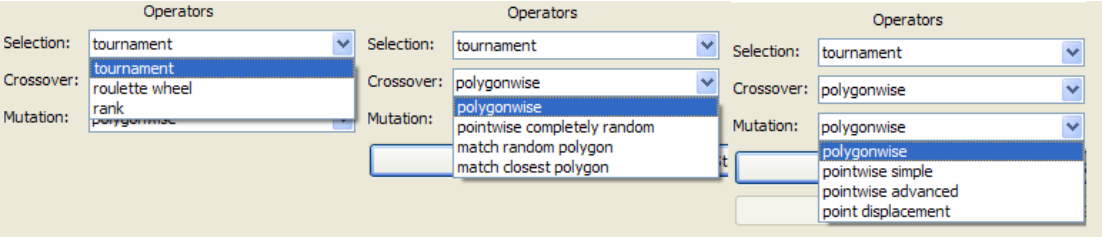


Slika 4.2 Grafičko sučelje

U gornjem lijevom kutu se podešavaju razni parametri za genetski algoritam, uključujući parametre za genetske operatore.

Ispod toga su uvjeti zaustavljanja algoritma. Ako se neki uvjet stavi na 0, neće se koristiti.

Imena genetskih operatora (Slika 4.3) program dobiva iz genetskog algoritma, pozivajući ga sa zastavicom *--send-operator-info*, kako ne bi morali biti upisani u korisničkom sučelju.



Slika 4.3 Genetski operatori

U polje *Picture* se odabere slika koju želimo dobiti, te se prikaže iznad na lijevoj strani. Na dnu namjestimo programe za crtanje i uspoređivanje.

Kada stisnemo *Start* pokreće se genetski algoritam sa svim namještenim parametrima koji natrag šalje podatke sučelju. Šalje se ime datoteke sa slikom najbolje jedinke, koja se prikazuje na desnoj slici kako bi se mogao pratiti napredak, te na desnoj strani trenutnu generaciju, dobrotu najbolje jedinke i brzinu računanja u *ips* (*iterations per second*) i trajanje jedne iteracije u milisekundama.

Sa tipkom *Stop* može se prijevremeno zaustaviti genetski algoritam.

5. Analiza rezultata

Napravljena su mjerenja učinkovitosti pojedinih genetskih operatora, kako bi se utvrdilo koji su najbolji i u kojoj kombinaciji najbolje rade.

Napravljene su tri skupine mjerenja:

1) kombinacija križanja i mutacije
2) selekcija
3) korištenje samo mutacije, bez selekcije i križanja

Slika na kojoj je vršeno ispitivanje je glava Leonardo da Vincijeve Mona Lise (Slika 5.1), veličine 180x195 piksela. Evoluirana slika sastoji se od 60 poligona, od kojih svaki ima 6 točaka.

Slika 5.1 Mona Lisa

Trajanje svakog ispitivanja je 100.000 iteracija. Sva ispitivanja traju približno jednako, zato što selekcija, križanja i mutacija traju puno manje od računanja dobrote. Računanje dobrote također ne ovisi o genetskim operatorima nego samo o veličini slike, broju poligona i broju točaka u poligonima.

Veličina populacije je 50 jedinki. Vjerojatnost mutacije je 0.3%, osim za ispitivanja samo s mutacijom, gdje se koristi više različitih vjerojatnosti.

5.1. Križanje i mutacija

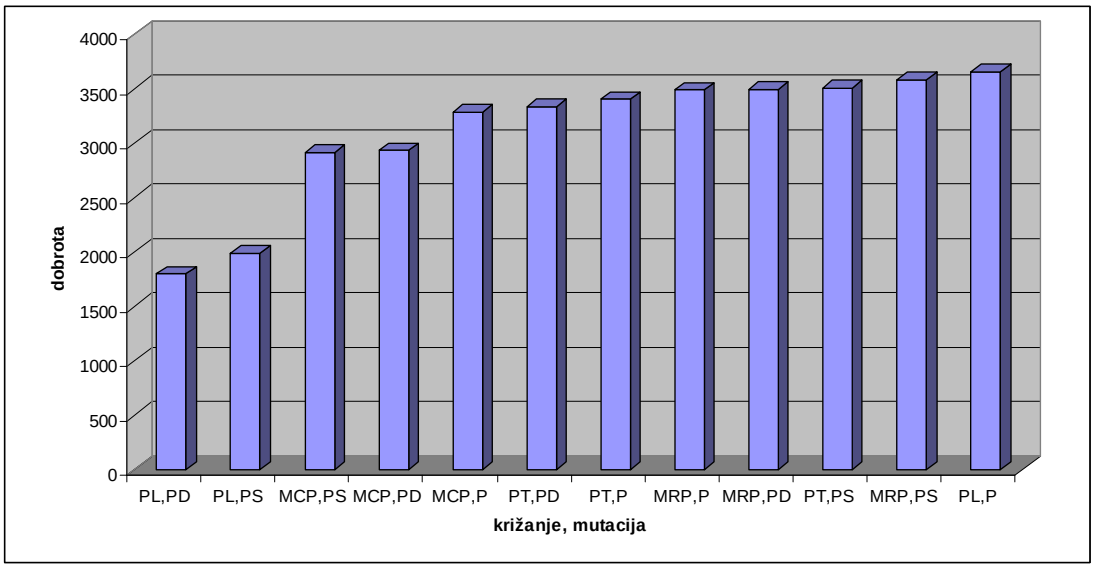
Koristila se samo jedna selekcija - *tournament*, a veličina turnira je 4.

Križanja koja su se koristila su: *polygonwise* (PL), *pointwise (PT), match random polygon* (MRP) i *match closest polygon* (MCP).

Mutacije koje su se koristile su: *polygonwise* (P), *pointwise simple* (PS), *point displacement* (PD).

Svako mjerenje izvedeno je 10 puta kako bi se smanjila slučajnost mjerenja.

Na slici 5.2 vidimo rezultate mjerenja sortirane po dobroti, manje je bolje.



Slika 5.2 Mjerenje učinkovitosti križanja i mutacije

Najbolja kombinacija je *polgonwise* za križanje, i *point displacement* za mutaciju sa dobrotom od 1795. Ako se mutacija zamjeni s *pointwise simple*, rezultati su dalje dobri - 1987. Sve ostale kombinacije su daleko lošije.

Na slici 5.3 može se vidjeti kako nakon 100.000 iteracija izgleda fenotip jedne lošije kombinacije (lijevo), te kako izgleda fenotip najbolje (desno). Slika i dalje nije prepoznatljiva, ali je očito da je desna bolja.

Slika 5.3 *pointwise, point displacement* (lijevo); *polygonwise, point displacement* (desno)

Polygonwise križanje, gdje se cijeli poligoni bez mijenja prenose na dijete, je bolje od ostalih zato što se u ostalim križanjima točke poligona previše izmiješaju, te dijete više nema dovoljne sličnosti s roditeljima.

Ako na sliku s N piksela želimo staviti jedan poligon od T točaka na točno određeno mjesto, tako da nasumično stavljamo cijeli poligon na sliku dok ne pogodimo, očekivani potreban broj stavljanja je N^T. Ako na istu sliku želimo staviti isti poligon, ali tako da stavljamo točku po točku dok ne pogodimo, očekivani broj pokušaja je $N \cdot T$, što je daleko manje.

Zbog tog razloga *polygonwise* mutacija, gdje se cijeli poligoni zamjenjuju s novim sasvim različitim poligonima, u kombinaciji s *polygonwise* križanjem je najlošija jer nemaju mogućnost preciznog mijenjanja poligona.

5.2. Selekcija

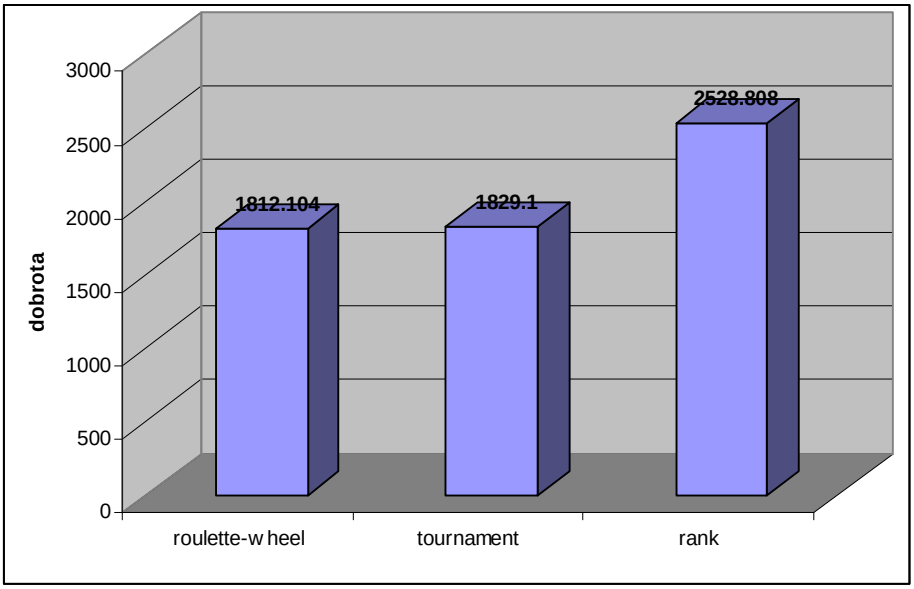
Selekcija se mjerila odvojeno od križanja i mutacije, zato što nema tolike međuovisnosti s njima. Mjerene selekcije su: *tournament*, *roulette-wheel*, *rank.* Veličina turnira je 4, a vjerojatnost u *rank* selekciji je 30%.

Za križanje i mutaciju odabrani su operatori koji su bili najbolji u prethodnom mjerenju:

križanje - *polygonwise,*

mutacija - *point displacement.*

Za svaku selekciju napravljeno je 10 mjerenja.



Slika 5.4 Mjerenja selekcija

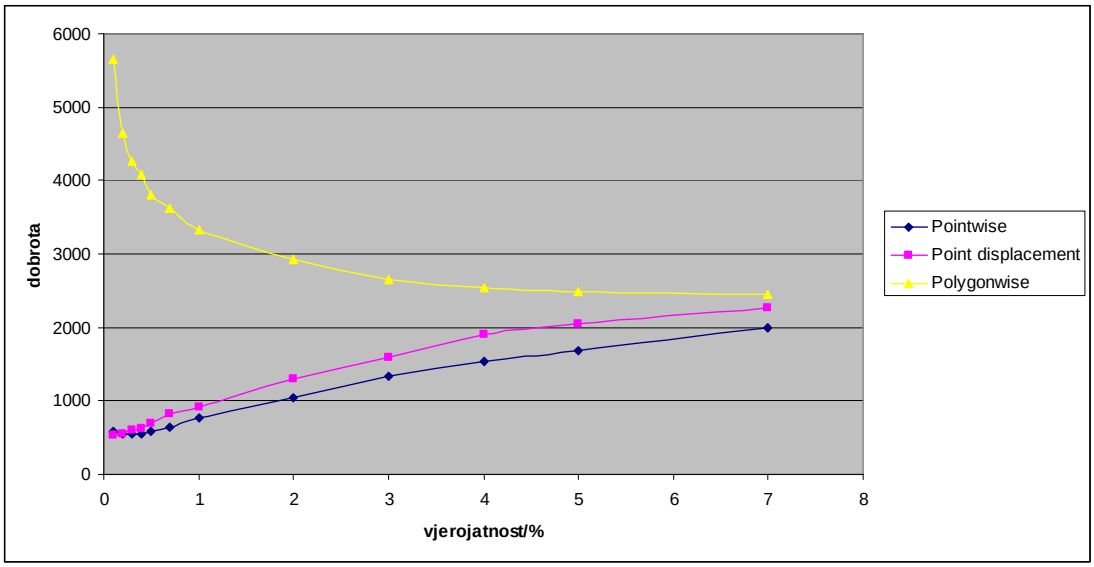
Na slici 5.4 iz rezultata mjerenja može se vidjeti da su *roulette-wheel* i *tournament* podjednako dobri, dok je *rank* selekcija 40% lošija.

5.3. Mutacija

U ovim ispitivanjima korištena je samo mutacija, bez selekcije i križanja. Zbog toga to ustvari nije genetski algoritam, nego više vrsta *random hill-climbera*.

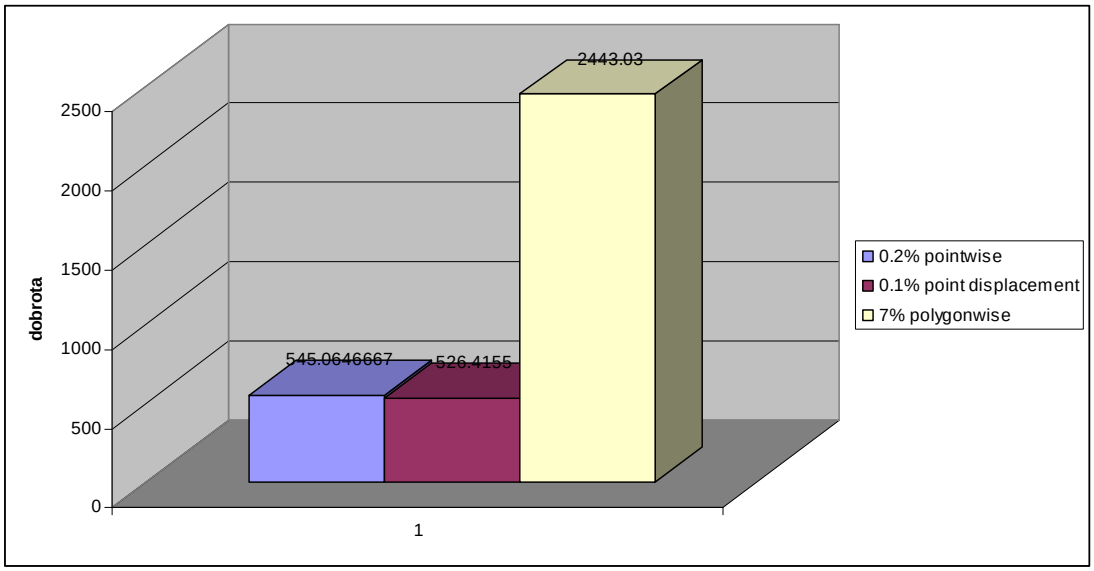
Rađena su mjerenja za 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 1, 2, 3, 4, 5 i 7 posto za svaki od operatora: *polygonwise*, *pointwise simple* i *point displacement*.

Svako mjerenje rađeno je 6 puta.



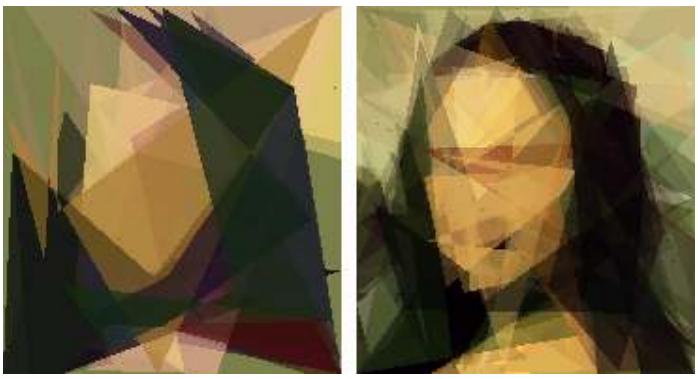
Slika 5.5 Mjerenje operatora mutacije

Na slici 5.5 vidi se da su *pointwise* i *point displacement* operatori najučinkovitiji, uz mali postotak mutacije (0.1-0.2%). Slika sveukupno ima 360 točaka, pa se uz malo veću vjerojatnost mutacije, događaju više mutacija odjednom, te uz dobru mutaciju, dogodi se i loša koja usporava napredak.



Slika 5.6 Najbolji rezultati mjerenja

Poligonwise mutacija na nižim vjerojatnostima radi jako loše, zato što slika ima samo 60 poligona, zbog čega je moguće da uopće ne dođe do mutacije. I na većim vjerojatnostima daleko je lošija od ostalih mutacija (Slika 5.6), zbog razloga navedenog u Ako.



Slika 5.7 Najbolje jedinke sa križanjem (lijevo) i samo mutacije (desno)

Uz isti broj iteracija, ispitivanja samo s mutacijom daleko su bolja nego ispitivanja gdje se koristi i selekcija i križanje (Slika 5.7).

6. Zaključak

U radu je napravljeno programsko rješenje genetskog algoritma koji iz zadane slike evoluira novu sliku, što sličniju početnoj, ali samo koristeći poluprozirne poligone. Pritom su korišteni razni genetski operatori.

Učinkovitosti pojedinih genetskih operatora eksperimentalno su izmjerene. Napravljeno je mjerenje svih kombinacija križanja (*polygonwise*, *pointwise*,, *match random polygon* i *match closest polygon*) s mutacijama (*polygonwise*, *pointwise simple*, *point displacement*) i utvrđena je najbolja kombinacija križanja *polygonwise*, gdje se prenose cijeli poligoni na dijete, s mutacijom *point displacement*, gdje se točka poligona pomakne za malu udaljenost od početne pozicije. Približno dobra mutacija s istim križanjem je *pointwise simple*, dok su sve ostale kombinacije dosta lošije. *Polygonwise* križanje s *polygonwise* mutacijom su najgora kombinacija zato što nijedan od njih nema mogućnost obavljanja manjih promjena na slici.

Selekcije *tournament* i *roulette-wheel* su podjednako dobre, dok je *rank* prilično lošija.

Uz korištenje samo mutacije, *point displacement* i *pointwise simple* su najbolji operatori i rade bolje uz manje vjerojatnosti mutiranja, dok je *polygonwise* daleko lošiji i radi bolje uz veće vjerojatnosti mutiranja.

Mjerenja sa svim genetskim operatorima (selekcija, križanje i mutacija) višestruko su lošija nego uz korištenje samo mutacije.

Literatura

[1] Roger, A.: Evolution of Mona Lisa, http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/ , 1.3.2009.

[2] Mitchell M.: „An Introduction to Genetic Algorithms“, MIT Press, 1999.

[3] Wong, T. C.: „Genetic Algorithms“, http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html , 2.6.2009.

[4] „Crossover (genetic algorithm)“, http://en.wikipedia.org/wiki/Crossover_(genetic_algorithm) , 6.6.2009.

[5] „Genetic Programming“, http://www.aerojockey.com/papers/lyapunovgp/node3.html , 7.6.2009.

Evolucija prikaza slike temeljenog na geometrijskim elementima

Sažetak:

Cilj ovoga rada je koristeći nekoliko poluprozirnih poligona prikazati neku sliku. To je postignuto genetskim algoritmom. Kromosomi koje genetski algoritam koristi sastoje se od boje pozadine, i nekoliko poligona, koji imaju svoju boju i prozirnost, te koordinate točaka. U programskoj implementaciji napravljeno je više genetskih operatora, i eksperimentnim metodama utvrđeno je da su podjednako dobre turnirska selekcija i *roulette-wheel* selekcija. Najbolje križanje je ono kod kojeg se na potomstvo prenose cijeli poligoni, bez promjena. Dobre mutacije su one koje točke poligona premještaju na malu udaljenost od početnog položaj, i one koje zamjene točku novom točkom.

Napravljeno je i mjerenje samo uz korištenje mutacije, i rezultati su daleko bolji nego sa selekcijom i križanjem.

Ključne riječi: genetski algoritam, vektorizacija

Evolution of presentation of images based on geometric elements

Abstract:

Goal of this paper is to display an image using only a few semi-transparent polygons. This is achieved using a genetic algorithm. Chromosomes which the genetic algorithm uses consist of a background color, a few polygons which have their color and transparency and point coordinates. The implementation has several genetic operators and using experimental methods it was concluded that tournament and roulette-wheel selection are equally efficient. Best crossover is the one which transfers polygons to the offspring intact. Good mutations are the ones which displace a point a small distance and the one which replaces a point with a new point.

A measurement using only mutation was also made, and the results are far better then the ones with selection and crossover.

Key words: genetic algorithm, vectorization