SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Završni rad br. 3309

Automatsko slaganje nepravilnih oblika uz pomoć evolucijskih algoritama

Jović Dino

Zagreb 2012/2013

Sadržaj

1. Uvod

Zadatak slaganja nepravilnih oblika je na ploču je jednostavan, međutim uvijek se može postaviti pitanje postoji li bolje rješenje, jer svaki oblik na ploči se može postaviti na skoro beskonačno mnogo položaja i svaki oblik može biti postavljen u odnosu na sve ostale oblike na također beskonačno mnogo položaja. Ako je neki postupak jako dobar za jedan određeni primjer nije poznato kako će se ponašati na drugačijim primjerima tj. ne postoji najbolje rješenje.

Optimizacijske probleme rješavamo kako bi dobili što bolje rješenje u razumnom vremenu.

Ovaj problem ima veliku upotrebu u industriji. Potrebno je izrezivati oblike različitih veličina i nepravilnog izgleda iz ploča od različitih materijala. Uz bolje korištenje materijala dobiva se manje otpada na kraju, tj. manje troškove za izradu istih proizvoda.

2. Opis problema

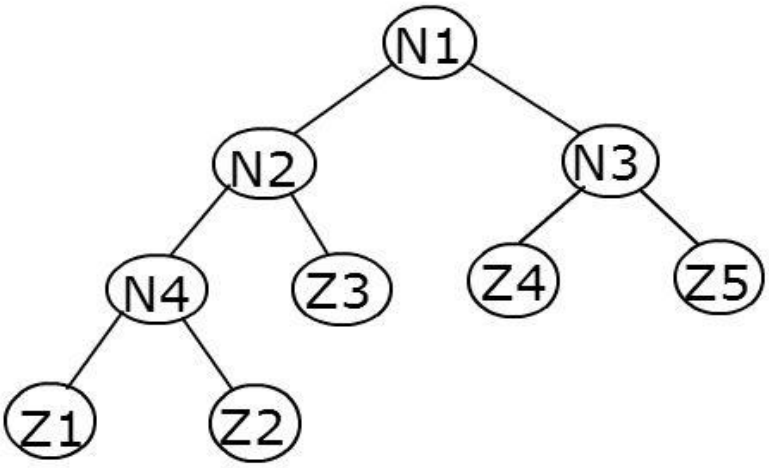
U ovome problemu prvo moramo moći odrediti koji je lik bolji, te prikazati taj podatak u obliku broja kako bi ga mogli uspoređivati s ostalim likovima.

U ovom radu za rješavanje navedenog problema uporabljeno je genetsko programiranje.

2.1. Genetsko programiranje

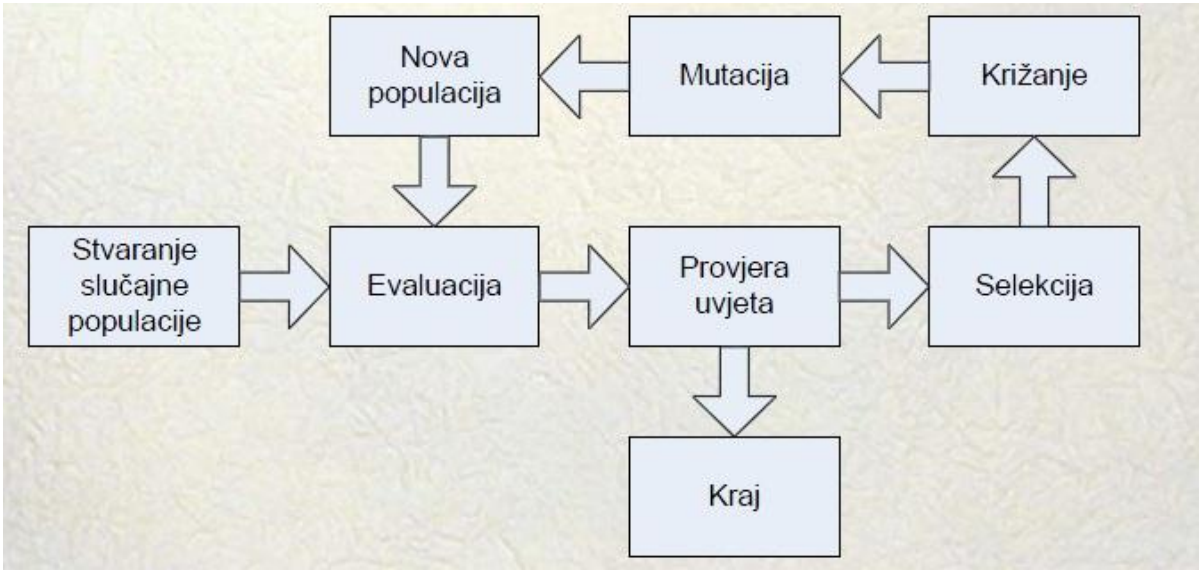
Genetsko programiranje je postupak simulacije evolucije, u kojoj se bolje jedinke križaju u nadi da će nastati novo bolja jedinka, dok lošije jedinke izumiru i ne utječu na jedinke u slijedećoj generaciji. Prvo se daje popis pojmova koji se koriste u genetskom programiranju:

1. Jedinka (stablo): je jedan algoritam koji se sastoji od završnih i nezavršnih znakova.
2. Završni znakovi (završni elementi ili listovi): dio algoritma koji je zadužen za izvršavanje konkretnih postupaka.
3. Nezavršni znakovi (nezavršni elementi ili čvorovi):dio algoritma koji mora imati barem 2 lista i/ili čvora kao djecu. Zadužen za odluku koje će se dijete izvršiti.
4. Populacija (generacija): skup jedinki koje se međusobno natječu, i iz kojih će nastati sljedeća populacija.
5. Evaluacija: postupak u kojem se svaka jedinka izvršava na ulaznom skupu podataka.
6. Dobrota: broj koji se postavlja na kraju evaluacije, a pokazuje koliko je neka jedinka dobra. Ovisno o problemu može biti bolja veća ili manja dobrota. U ovom radu je bolja dobrota većeg iznosa.
7. Križanje: postupak u kojem se odabiru dvije jedinke i uzme se dio svake te napravi nova jedinka koja će biti u sljedećoj generaciji.
8. Mutacija: slučajna promjena na jedinci.
9. Selekcija: postupak odabira jedinki koje će se križati da dobijemo sljedeću generaciju.



Slika 2.1.1. Jedinka

Ovdje je prikazan izgled jedinke općenito u genetskom programiranju. Svaki element označen slovom N je nezavršni element i mora imati barem dvoje djece, dok je svaki element označen slovom Z završni element i ne smije imati djece.



Slika 2.1.2. Dijagram toka genetskog programiranja.

Slika preuzeta iz [9] u kojoj je opisan postupak genetskog programiranja. Prva generacija je slučajna, a zatim nastaju nove generacije koje se poboljšavaju sve dok se ne zadovolji uvjet zaustavljanja.

Uvjet zaustavljana može biti dostignuta određena dobrota, proteklo vrijeme evolucije, određeni broj generacija ili broj generacija bez nove najbolje jedinke.

Rješenje cijelog algoritma je najbolja jedinka, koja će se dalje moći koristiti na svim ulaznim podacima uključujući i one na kojima se nije evoluirala.

2.2. Opis rješavanja problema

Rješavanje problema započinje tako da učitamo podatke o karakteristikama algoritma iz konfiguracijske datoteke. Kada imamo učitane podatke stvara se novi objekt razreda TreeFactory koji je zadužen za sva stvaranja završnih i nezavršnih elementa stabla kao i njihov međusobni odnos, te osigurava da se stabla ne krše uvjete definirane u karakteristikama algoritma. Poslije toga učitavamo u algoritam drugu ulaznu datoteku u kojoj se nalaze podaci o dimenzijama ploča te o broju i obliku poligona na tim pločama.

Oblike postavljamo na ploču tako da se svaki oblik prvo rotira za slučajan kut između 0 i 360 stupnjeva ,a zatim translatira lijevo i dolje za njihove minimalne X i Y koordinate. Tada se dobije oblik koji je pozicioniran uz koordinatne osi X i Y, te ga na kraju translatiramo desno za slučajnu udaljenost između 0 i maksimalne širine ploče smanjene za maksimalnu X koordinatu toga oblika. Tako analogno pomaknemo i oblik za gore.

Svaka jedinka u generaciji se pojedinačno ocjenjuje. Kada je završena evaluacija odabiru se jedinke za križanje i stvara se nova generacija. Neke jedinke se mutiraju i na kraju se najbolji iz prethodne generacije stavi u novu generaciju.

2.3. Dobrota

Dobrota je racionalan broj koji opisuje koliko je dobro potencijalno rješenje. Za računanje dobrote nam trebaju ulazni podaci i lista svih ploča nakon izvršavanja algoritma.

Prva varijabla koju računamo je broj linija svih oblika koji se sijeku u svakoj ploči, zatim za svaku ploču podijelimo taj broj s brojem oblika u ploči. Tu normalizaciju radimo jer ako je neki algoritam na nekoj ploči s npr. 3 oblika imao isti broj presjeka kao neki drugi algoritam na ploči s npr.10 oblika, iako su postigli isti rezultat prvi algoritam je lošiji. Nakon normalizacije zbrajaju se rezultati svake ploče i taj broj pomnožimo s 10 kako bi povećali razlike između ostvarenih rezultata. Idealni će algoritmi (oni u kojima se oblici ne sijeku) dobiti rezultat nula tj. broj koji smo dobili zapravo govori koliko je neki algoritam loš. Iz tog razloga odabire se konstanta

velikog iznosa od koje se oduzima dobivena vrijednost. U ovom radu ta konstanta je postavljena na 1000.

Algoritmi koji daju oblike raspoređene tako da nema sječenja imat će dobrotu 1000. U tom slučaju dodajemo drugu varijablu koja opisuje koliko je prostora na ploči ostalo slobodno. Algoritmi koji na nekim pločama imaju sječenja a ne nekima nemaju se također izbacuju iz drugog dijela određivanja dobrote jer one ploče na kojima se oblici sijeku mogu jako poboljšati dobrotu zato jer im treba manje prostora nego normalnim algoritmima jer oni isti prostor koriste više puta, što u stvarnosti nije moguće napraviti. Za algoritme čija se rješenja ne sijeku na svakoj ploči tražimo najmanji pravokutnik koji obuhvaća sve oblike u potpunosti, a to radimo tako da jednostavno prođemo kroz svaki oblik i gledamo minimalne i maksimalne koordinate za obje koordinatne osi i za svaki od tih ekstrema pamtimo globalnu vrijednost na toj ploči i na kraju dobijemo površinu tog pravokutnika kao umnožak razlika minimalne i maksimalne udaljenosti po svakoj koordinatnoj osi. Nakon toga se računa koliko je ostalo slobodne ploče u postocima od cijele ploče kako bi osigurali da različite veličine ploča jednako utječu na dobrotu rješenja. Za svaku ploču te vrijednosti se zbrajaju i dijele s brojem ploča. Ovu normalizaciju radimo zbog toga da osiguramo mogućnost usporedbe rezultata između algoritma koje smo ispitali na različitom broju ispitnih ploča. Dobivena vrijednost dodaje se na prethodnu vrijednost dobrote.

Kratko objašnjene značenja dobrote F.

$F < 1000$, Oblici se sijeku, loša rješenja.

$F = 1000$, Iskorištena je cijela ploča

$1000 < F < 1100$, Oblici se ne sijeku, rješenja koja tražimo.

$F = 1100$, Nema oblika na niti jednoj ploči, ploče su prazne

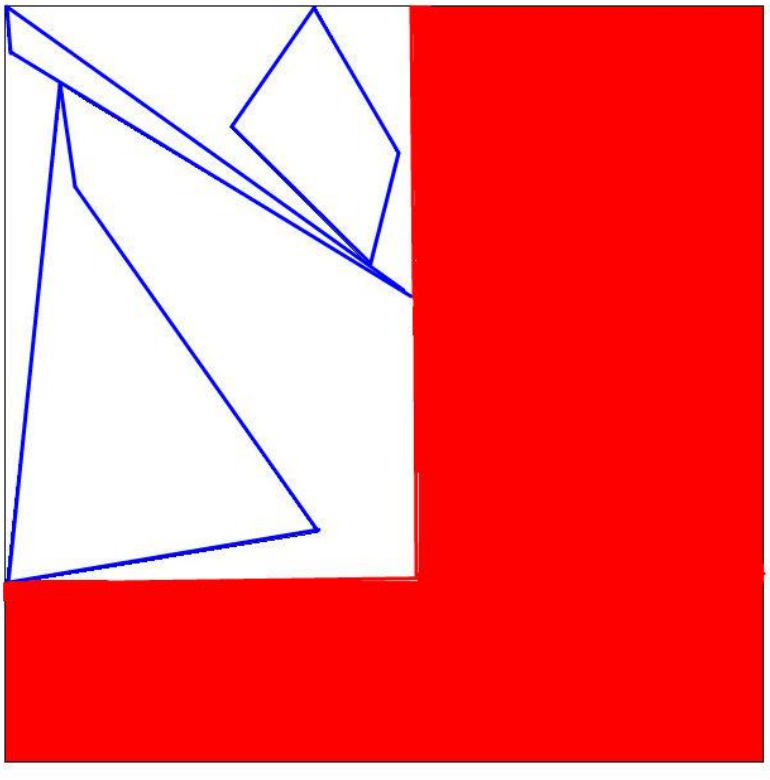
$F > 1100$, Nemoguće, ostalo je više ploče nego što smo imali na početku.

```
Best fittnes : 1055,49361594601
    TU
        TU
        TU
    IMSL
        TD
        TD

IMSR
    TU
    TD
```

Slika 2.3.1. Primjer rješenja i njegovog rezultata.

Na slici 2.3.1. prikazan je jedan primjer rješenja i njegova dobrota koju dobivamo nakon izvršavanja na ploči prikazanoj na sljedećoj slici. U svakom redu je prikazan jedan element stabla a u stupcima su prikazane njihove dubine. Element najviše lijevo je korijen stabla. Elementi iznad korijena su djeca koja rade promjene na ploči na temelju kojih se odlučuje koje dijete od djece ispod njega će se izvršiti. Iznos dobrote je dolje prikazan crvenim djelom ploče, što je iskoristiv dio ploče nakon obrade, a prema dobroti vidimo da je to malo više od 55%. Ovaj primjer za svaki oblik gleda koliko ima mjesta desno za taj oblik, djeca za odluku su translacija gore i drugi nezavršni znak koji neovisno o uvjetima radi pomak dolje. Ako ima više mjesta nakon prvog djeteta izvršiti će se pomak gore, a ako ima nakon drugog djeteta onda pomak dolje.



Slika 2.3.2. primjer izvršavanja rješenja na jednostavnoj ploči.

2.4. Ulazni podaci

Ulazni podaci su ploče i oblici koji su na pločama. Svaka ploča mora imati svoje dimenzije, a ploče mogu biti samo pravokutne. Svaka ploča mora imati i popis oblika na njoj. Oblici mogu biti samo mnogokuti bez rupa u sebi i bez zakrivljenih linija. Pretpostavlja se da će korisnik zadati oblike tako da se ne sijeku sami sa sobom te da će dimenzije oblika biti manje od dimenzija ploče. Oblici se zadaju kao niz točaka za koje se pretpostavlja na nisu na istom pravcu.

Broj ploča koje se može učitati je neograničen, broj oblika je također neograničen ali se pretpostavlja da korisnik neće učitati toliko oblika da oni ne stanu na ploču.

3. Opis programa

3.1. Korištene programske strukture

Koriste se 4 kontrolna razreda, zadužena za izvršavanje zadataka genetskog programiranja poput odabira jedinki, njihovog križanja i mutacije. Elementi stabla sastoje se od 3 apstraktne razreda, i 14 konkretnih od kojih su 4 nezavršne i 10 završnih koji rade poslove iz domene primjene.

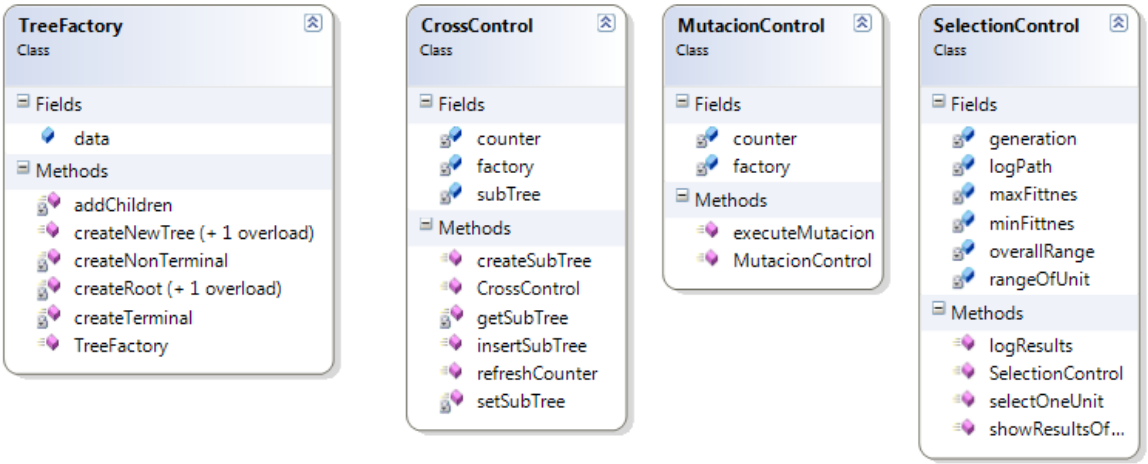
3.1.1. Razredi operatora

Kontrola odabira izvodi se tehnikom ruletnog kotača.

Svaka jedinka u generaciji dobiva svoj odjeljak na ruletnom kotaču. Što je jedinka bolja njezin će odjeljak biti veći te zbog toga će imati veću šansu da bude odabrana.

Mutacija se provodi slučajnom zamjenom jednog elementa stabla drugim slučajno odabranim elementom. Mutacija u većini slučajeva pogoršava dobrotu ali mora postojati zbog dodavanja novog genetskog materijala koji nije bio prisutan u trenutku stvaranja početne generacije.

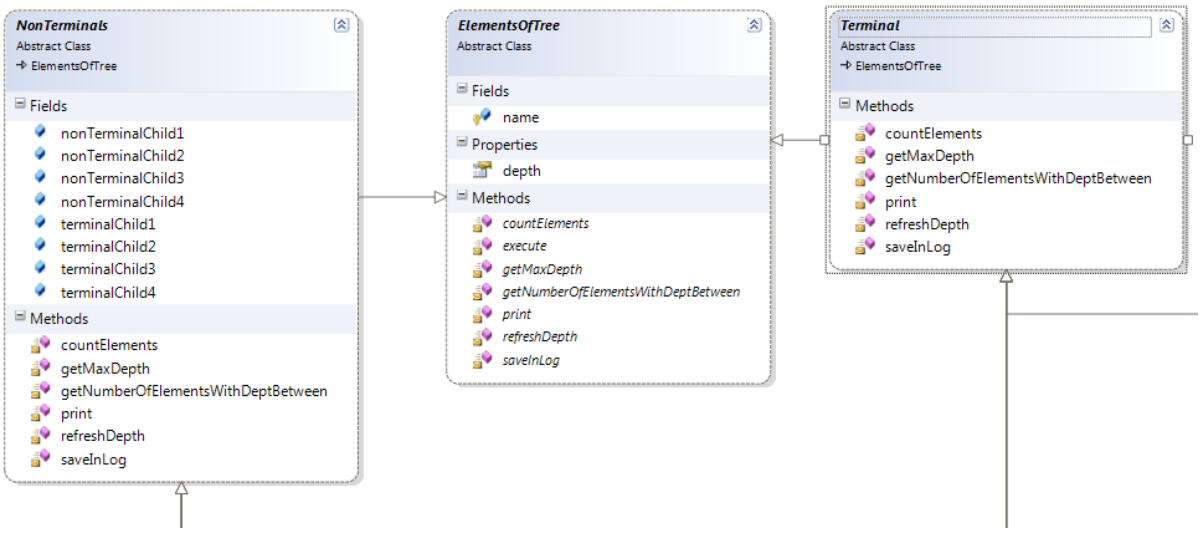
Tvornicu stabala (eng: *Tree Factory*) je razred koji koristimo kao centralizirano mjesto za kreiranje i modifikaciju svih stabala.



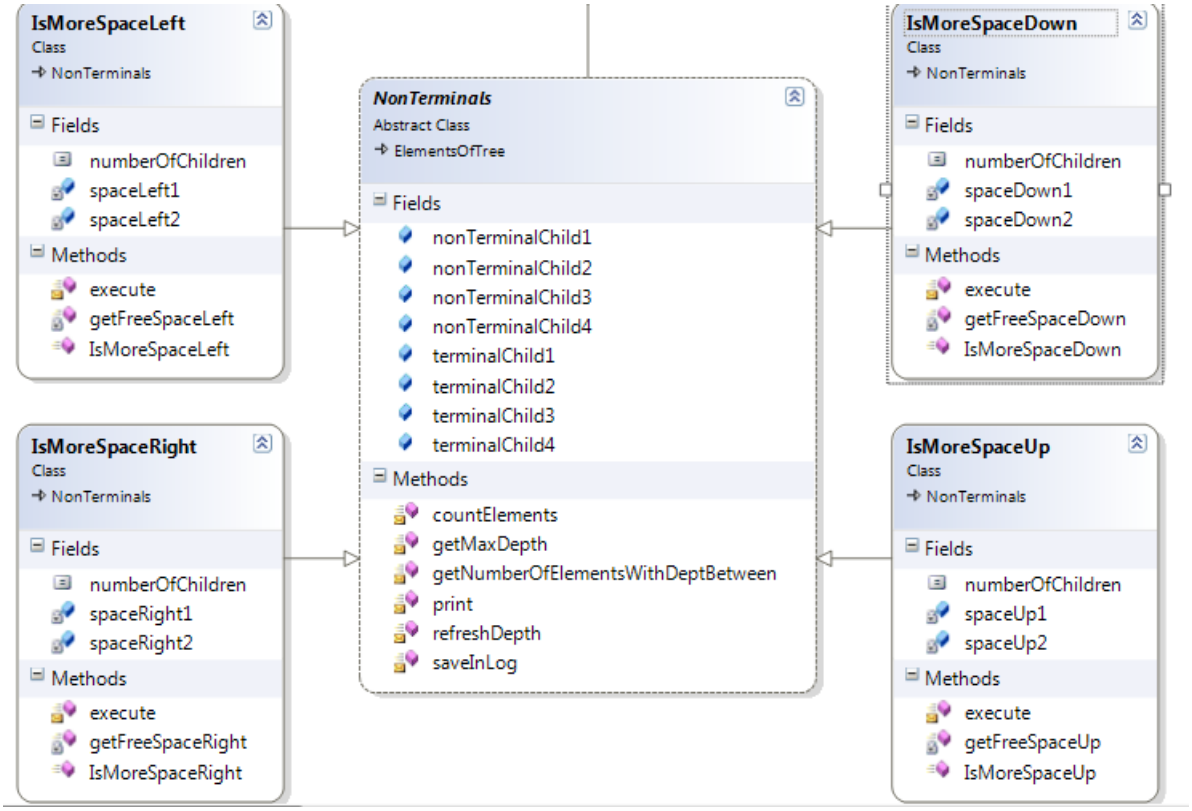
Slika 3.1.1.1. Razredi operatora.

3.1.2. Elementi Stabla

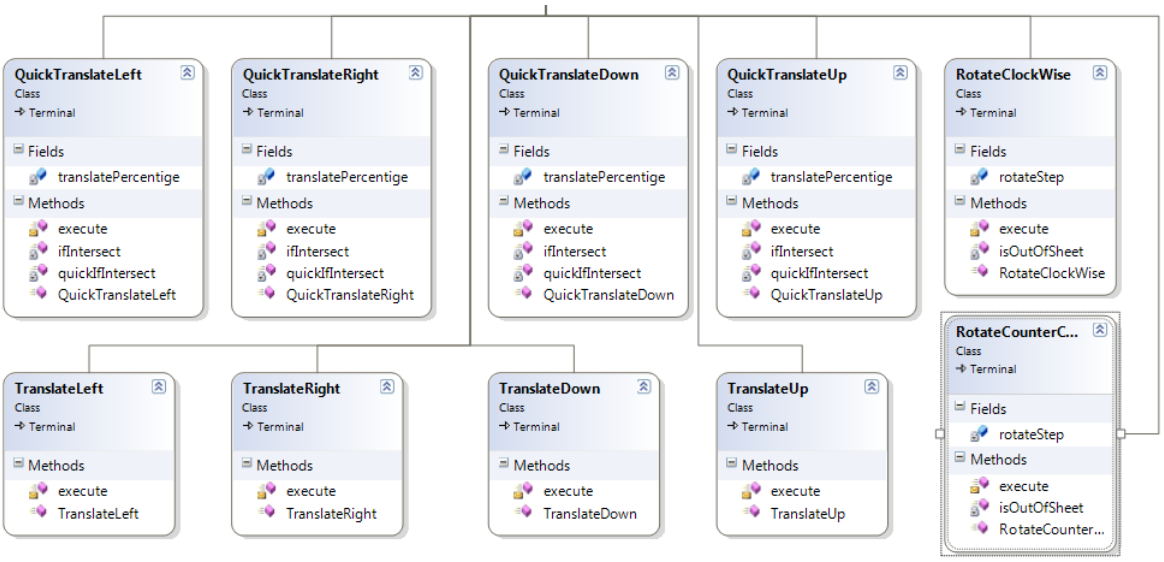
Elementi stabala koriste se za kreiranje algoritama i njegovo izvršavanje.



Slika 3.1.2.1. Apstraktni razredi algoritma.



Slika 3.1.2.2. Implementacija nezavršnih elemenata algoritma.

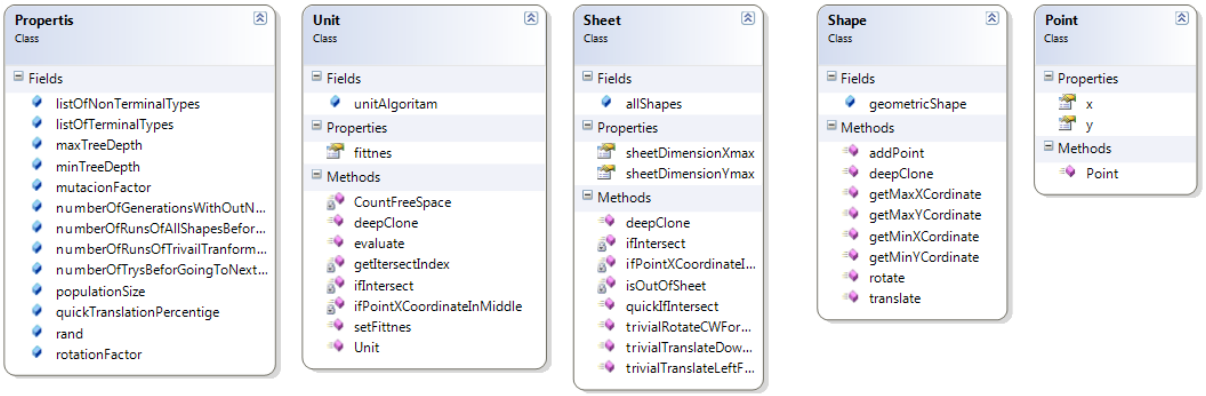


Slika 3.1.2.3. Implementacija završnih elemenata algoritma.

3.1.3. Spremanje podataka

Podatke spremamo u 5 različitih razreda koje su:

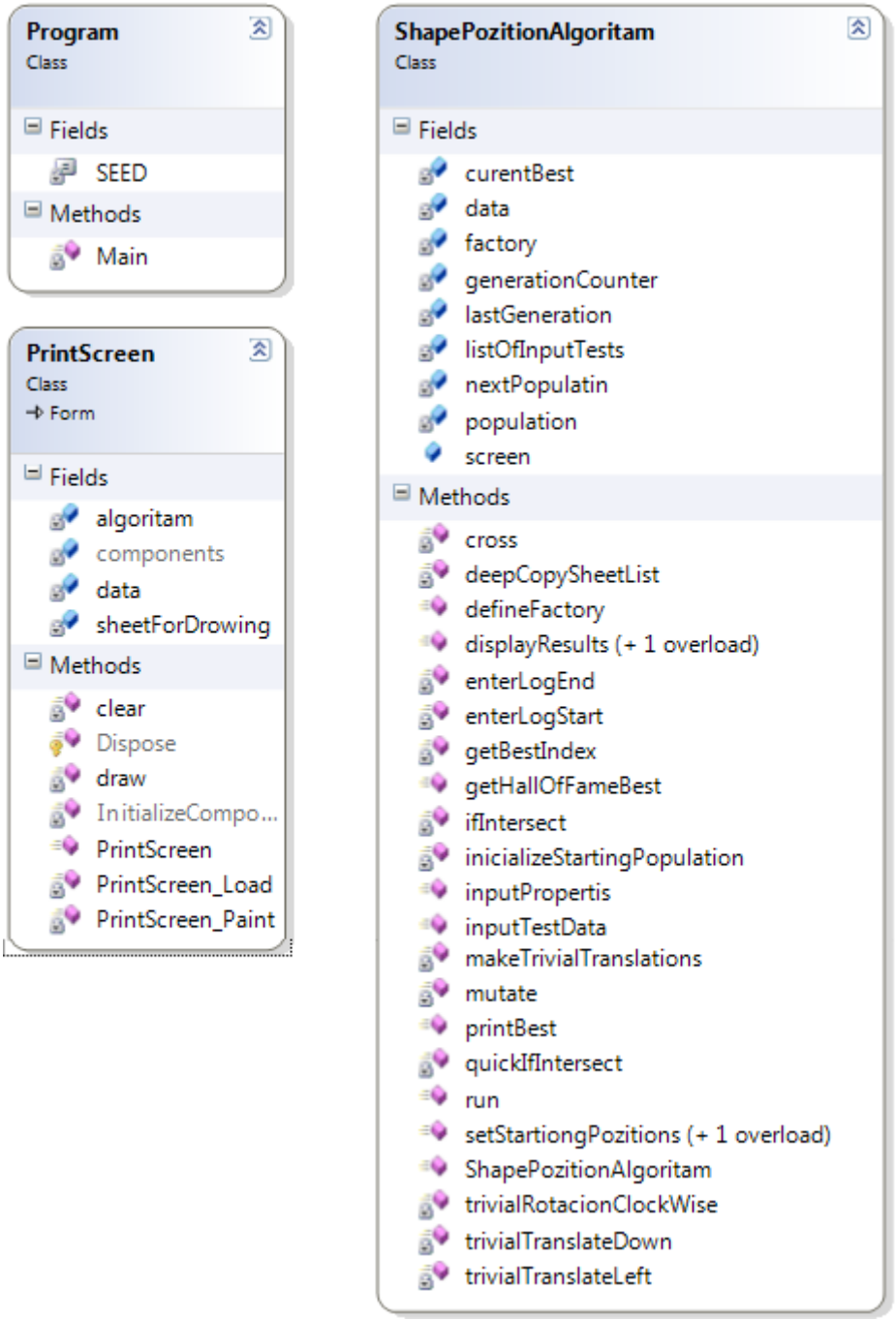
1. Točka – ima samo svoju X i Y koordinatu.
2. Oblik – ima listu točaka i metode koje mu omogućuju dodavanja novih točaka, izvršavanje translacije i rotacije, traženje minimalni i maksimalnih koordinata na obje osi.
3. Ploča – ima listu oblika, te svoje maksimalne i minimalne dimenzije, a može gledati dali se oblici na njoj sijeku, dali su izvan nje i izvršavati trivijalne transformacije oblika na njoj.
4. Jedinka – ima jedno stablo i svoju dobrotu, te metode za računanje i postavljane svoje dobrote i evaluaciju sebe nad pločama.
5. Svojstva – u sebi čuva konstante koje se učitavaju prilikom inicijalizacije i koristi se u konstruktorima većine ostalih razreda, također sadrži i jedan objekt razreda *Random* koji je zadužen za sve slučajne vrijednosti koje treba bilo koji dio programa.



Slika 3.1.3.1. Razredi za spremanje podataka.

3.1.4. Glavne strukture programa

Glavne strukture programa koristimo za povezivanje svih ostalih dijelova programa, usklađivanje njihovog rada i komunikaciju s korisnikom.



Slika 3.1.4.1. Glavni razredi programa.

3.2. Logika

Karakteristike algoritma koje moramo na početku odrediti su:

1. Minimalna dubina stabla: mora biti cijeli nenegativan broj.

2. Maksimalna dubina stabla: mora biti cijeli ne negativan broj, te mora biti veća ili jednaka minimalnoj dubini stabla. Što je maksimalna dubina stabla veća možemo dobiti kompleksnije uvjete ispitivana pozicije svakog oblika ali s cijenom velikog usporavanja programa.

3. Faktor mutacije: mora biti racionalan broj između 0 i 1. Služi za dodavanje genetskog materijala koji nije bio prisutan na početku algoritma. Prenisko postavljen ne dodaje dovoljno novog genetskog materija pa dolazi do brze stagnacije, previsoko postavljen može poništiti rezultate evolucije s previše slučajnog genetskog materijala.

4. Veličina populacije: mora biti cijeli broj veći od nule. S malom populacijom ubrzavamo evolucije ali koristimo puno manje genetskog materijala što može dovesti do stagnacije u lokalnim maksimumima, te se jako oslanjamo na faktor mutacije da nas izvede iz njih, a s velikom populacijom usporavamo evoluciju zbog dužeg izvođenja te i zbog mogućnosti ponavljanja više istih jedinki.

5. Uvjet zaustavljanja: broj 1,2,3 te cijeli broj veći on nule
   4.1. Maksimalan broj generacija: program se izvodi dok ne prođe unaprijed predviđen broj generacija.
   4.2. Maksimalan broj evaluacija: slično kao i maksimalan broj generacija samo što vrijeme izvršavanja programa ne ovisi o veličini populacije.
   4.3. Maksimalan broj generacija bez poboljšanja: svaki puta kada dobijemo novo najbolje rješenje program se produžuje, s njime skraćujemo vrijeme izvođenja programa i dobivamo lošije rješenje. Ako je dobro postavljen možemo puno skratiti trajanje a rješenje malo oslabiti. Ako je loše postavljen možemo ili ne skratiti trajanje ili dobiti katastrofalno loše rješenje.

6. Korak rotacije: mora biti racionalan broj između 0 i 360. Koristi se kod rotacije oblika što je korak manji dobivamo preciznija rješenja ali usporavamo program.

7. Postotak kratke translacije: mora biti racionalan broj između 0 i 1 koristi je zbog omogućavanja rotacije u kasnijim koracima algoritma kada se zbog blizine oblici više ne mogu rotirati, što je postotak manji dati će manje mjesta za ostale transformacije ali će imati veću šansu da se izvrši, ako je postotak prevelik češće će dolaziti do sječenja oblika nakon kratke translacije .

Treba odrediti koje sve završne i nezavršne elemente stabla ćemo koristiti. Nezavršni elementi stabla imaju kao svoju djecu najviše 4 završna elementa ili 4 nezavršna elementa ili njihove kombinacije. Nezavršni elementi prvo izvršavaju prvo dijete pa s njega uzimaju željeni podatak, zatim izvršavaju drugo dijete pa s njega uzimaju željeni podatak te ih uspoređuju. Ako je željena usporedba koja je drugačija za svaki nezavršeni element istinita onda se izvršava 3 dijete, a ako je usporedba laž onda se izvršava 4 dijete. Svi elementi se izvršavaju samo za jedan oblik u jednom trenutku i sve se operacije izvršavaju samo nad tim oblikom dok se ostali oblici gledaju samo u trenucima kada se provjerava postoje li križanja i kada se gleda koliko ima slobodnog mjesta. Svaki element prije izvršavanja dobije ploču i indeks oblika koji će obrađivati, a taj oblik se zove zadani oblik.

.

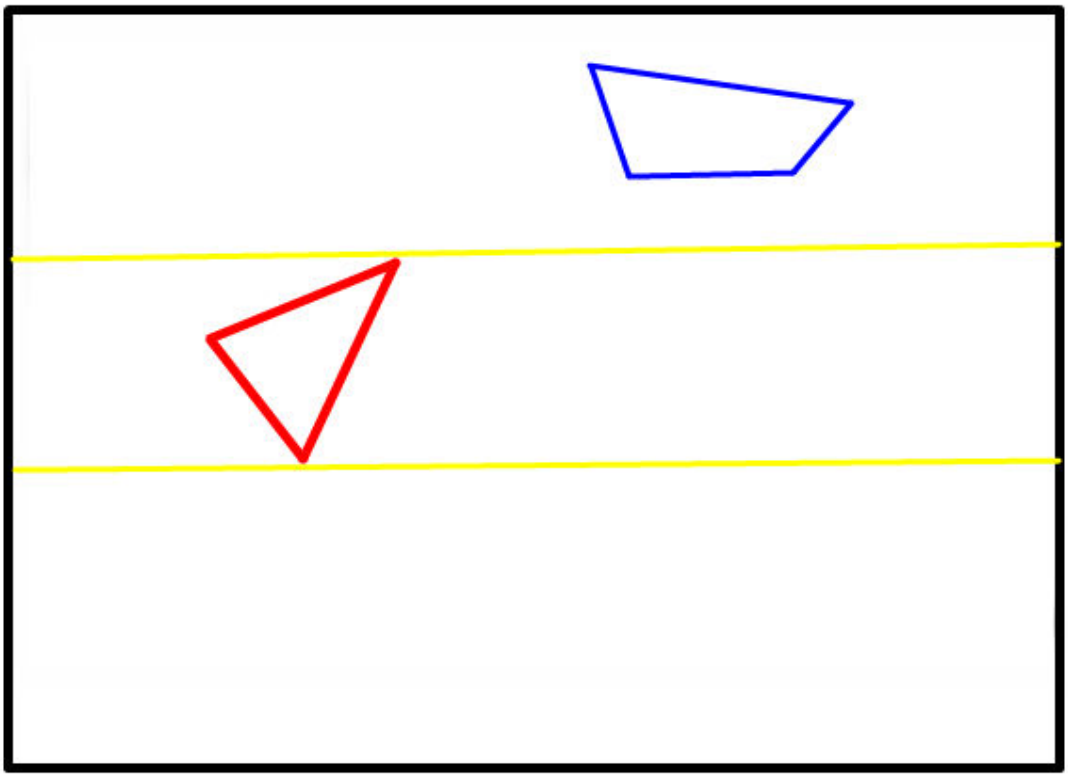
3.2.1. Translacije

1. TD (eng: translate down) translacija dolje. Gleda za sve oblike osim ulaznog da li se nalaze u vertikalnoj pruzi omeđenoj s minimalnom i maksimalnom X koordinatom ulaznog oblika, ako se neki nalazi i njegova maksimalna Y koordinata je ispod minimalne Y koordinate ulaznog oblika pamti se ta udaljenost između njih i gleda se tako za svaki oblik i traži se minimalna udaljenost. Na kraju se oblik pomakne za tu minimalno udaljenost dolje ako ne postoji niti jedan oblik ispod ulaznog oblika on se pomakne do granice ploče.
2. TL (eng: translate left) translacija lijevo. Radi analogno translaciji dolje samo su zamijenjene X i Y koordinatne osi.
3. TU (eng: translate up) translacija gore. Radi analogno translaciji dolje, s razlikom da se gleda udaljenost između vrha oblika i vrha ploče.
4. TR (eng: translate right) translacija desno. Radi analogno kao translacija gore s razlikom da gleda točku s najvećom X koordinatom i desnim krajem ploče.

Crveni oblik – oblik koji promatramo

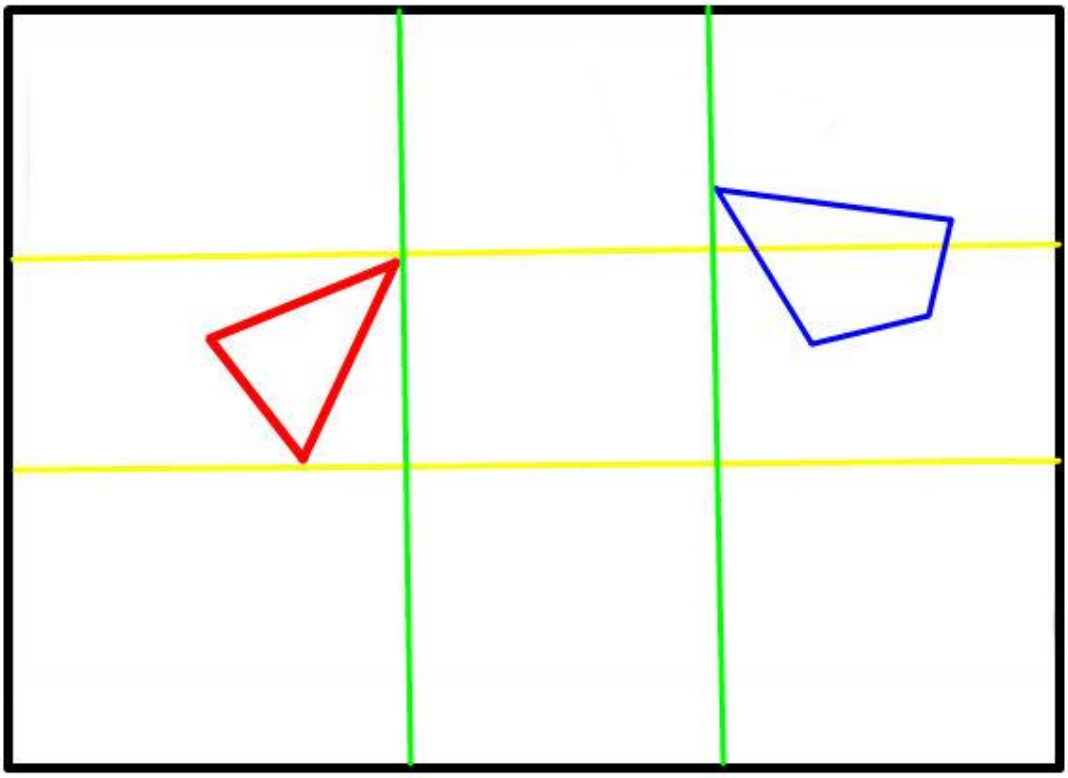
Plavi oblici – oblici koji smetaju

Promatramo na primjeru translacije desno, analogno vrijedi za translacije lijevo, gore i dolje. Kratke translacije rade drugačije i biti će objašnjene kasnije.



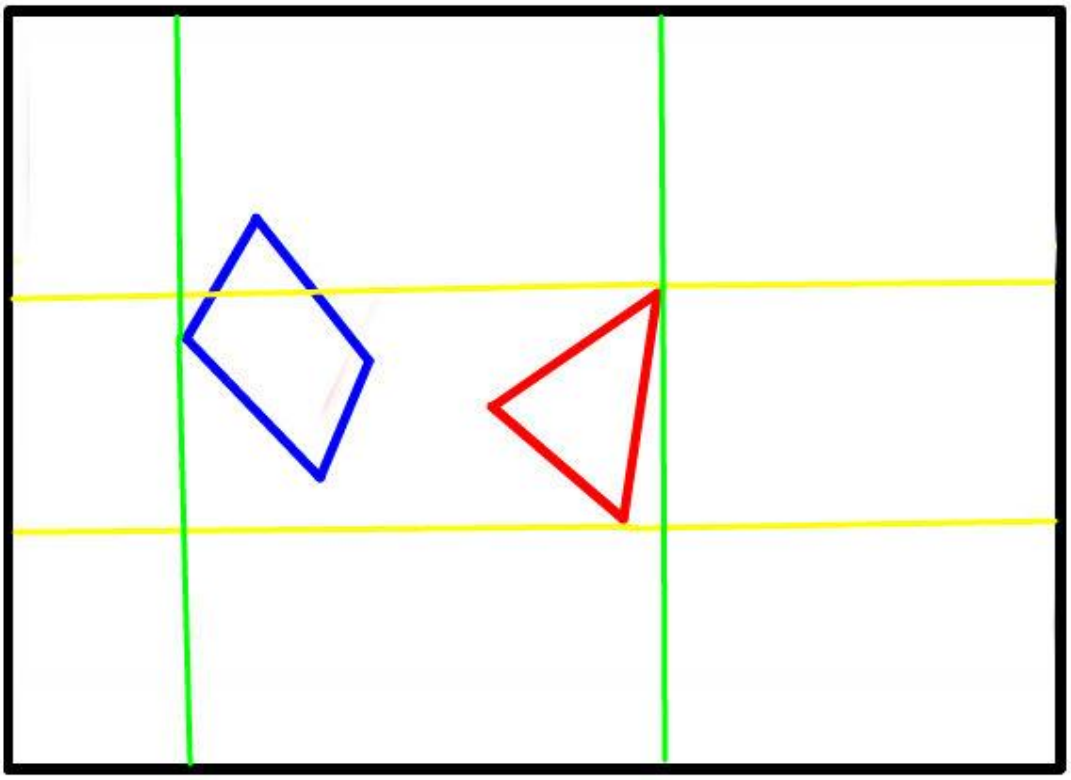
Slika 3.2.1.1. Translacija: drugi objekt ne smeta.

Žute linije označavaju minimalne i maksimalne koordinate na crvenom obliku. Ako je minimalna Y koordinata plavog oblika iznad maksimalne koordinate crvenog oblika onda plavi oblik ne smeta. Analogno vrijedi i ako se plavi oblik nalazi ispod crvenoga. Ako mu drugi oblici ne smetaju crveni se može pomaknuti desno do kraja ploče.



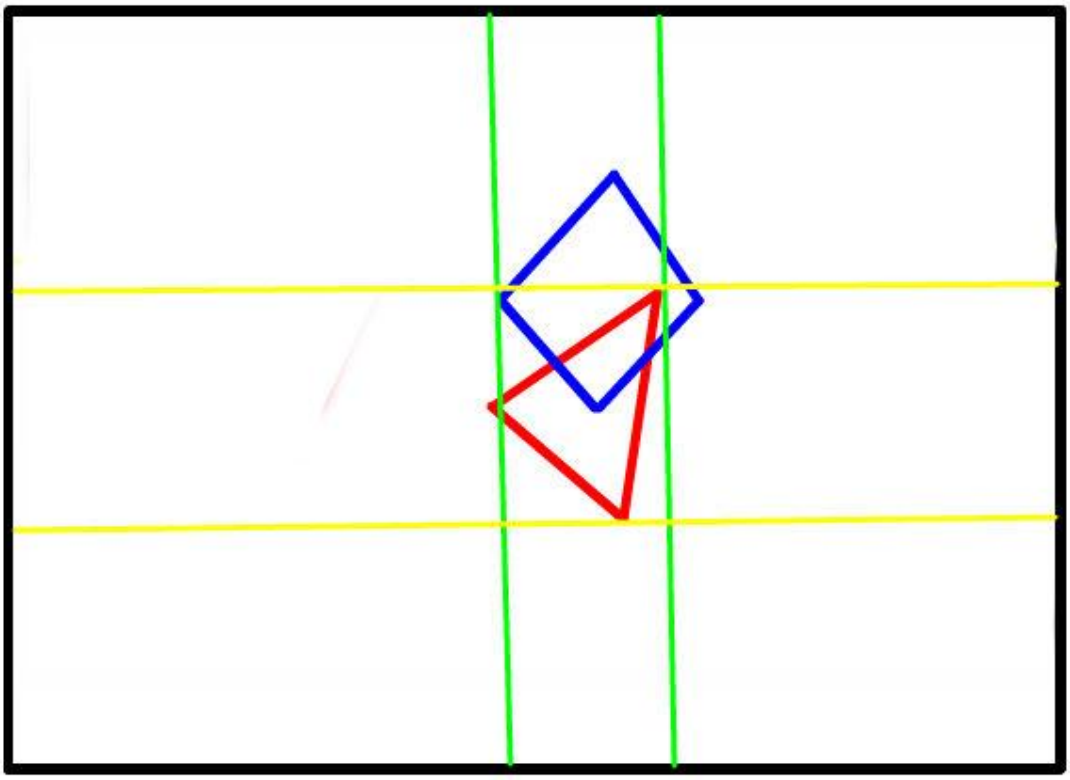
Slika 3.2.1.2. translacija: drugi oblik smeta.

Ako utvrdimo da po visini plavi oblici smetaju crvenome onda gledamo maksimalnu X koordinatu crvenog oblika i minimalnu X koordinatu plavog oblika. Obje koordinate su prikazane zelenim linijama. Ako je koordinata plavog oblika veća od one crvenog to znači da je plavi više desno tj. da će smetati. Tada smijemo crveni oblik pomaknuti za udaljenost između dviju zelenih linija.



Slika 3.2.1.3. Translacija: smeta po visini, po širini ne.

Isto kao na prethodnoj slici samo je gledana koordinata plavog oblika manja od gledane koordinate crvenog oblika. Crveni oblik se može pomaknuti desno do granice ploče. Zbog činjenice da oblici imaju konačne dimenzije možemo Može se dogoditi preklapanje kao što je prikazano na slijedećoj slici.



Slika 3.2.1.4. preklapanje.

Ovakvo preklapanje uzima se u obzir u izračunu dobrote jedinke.

Ako bi imali primjer u kojem imamo sva 4 plava objekta s prethodnih slika na jednoj ploči u istim odnosima s crvenim oblikom onda bi pomaknuli crveni oblik kao na slici 4.2.1.2. te zanemarili sve ostale.

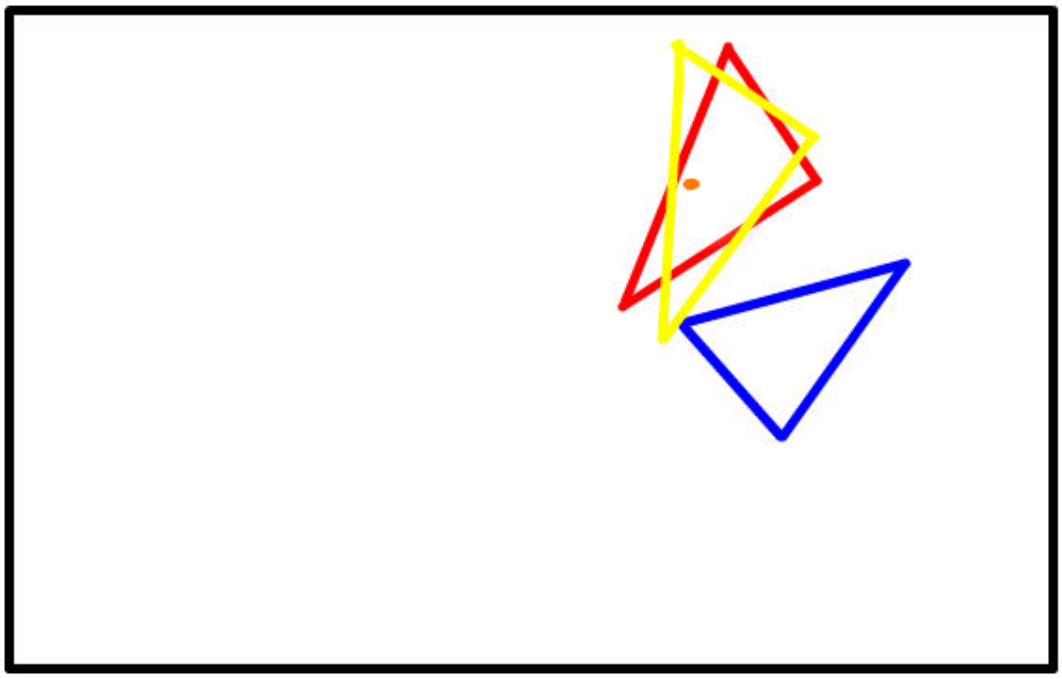
3.2.2. Rotacije

Rotacije svakog elementa se izvode oko aritmetičke sredine svih njegovih točaka koje na slikama označavamo s narančastom točkom. Imamo samo dvije rotacije:

1. RCW (eng: rotate clock wise) rotiraj u smjeru kazaljke na satu. Rotira ulazni oblik za rotacioni korak, nakon rotacije pregledava se da li se oblik sječe s nekim drugim oblikom ili je izvan granica ploče, ako je istina neki od ta 2 uvjeta vrača se 1 rotacioni korak u natrag i prekida se rotiranje ako nije rotira se za još jedan korak. Ako ukupna rotacije prijeđe 360 stupnjeva ispituje se dali je ploča šira ili viša i uzme se manja dimenzija i rotira se za onaj kut koji dovede oblik u poziciju u kojoj ima minimalnu visinu ili širinu ovisno o dimenzijama

ploče. U svakom koraku rotacije se računa širina i visina te ako je manja od najmanje do sada pamte se njihovi iznosi kutova na kojima su postignuti.

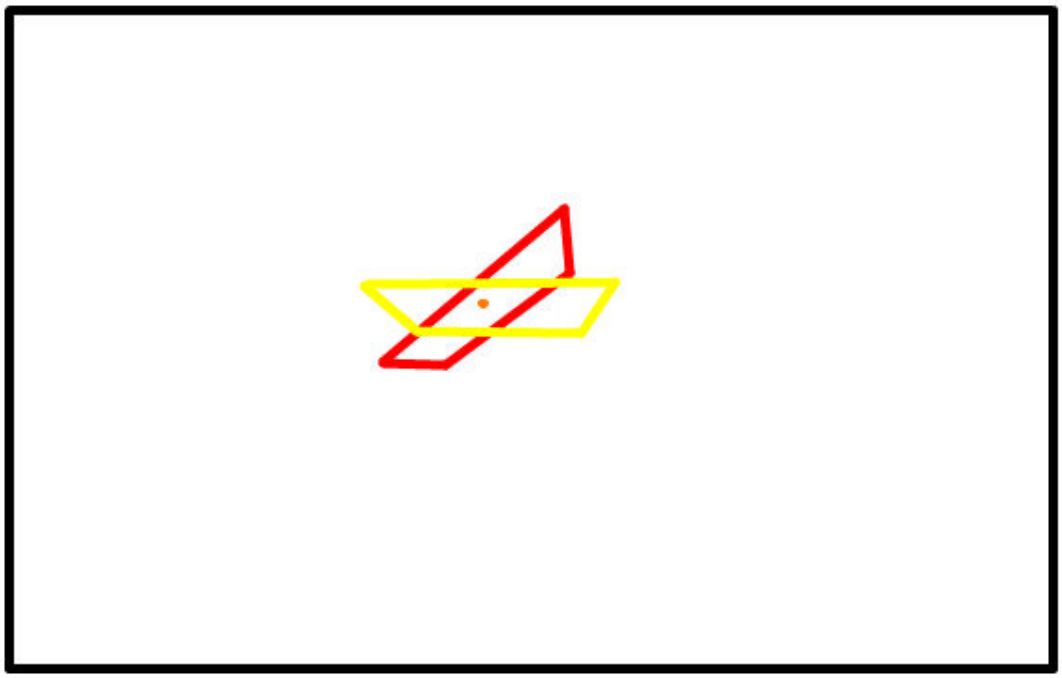
2. RCCW (eng: rotate counter clock wise) rotiraj u smjeru suprotnom od smjera kazaljke na satu. Rotira ulazni oblik za rotacioni korak, nakon rotacije pregledava se dali se oblik sječe s nekim drugim oblikom ili je izvan granica ploče, ako je istina neki od ta 2 uvjeta vrača se 1 rotacioni korak u natrag i prekida se rotiranje ako nije rotira se za još jedan korak. Ako ukupna rotacije prijeđe 360 stupnjeva ispituje se dali je ploča šira ili viša i uzme se manja dimenzija i rotira se za onaj kut koji dovede oblik u poziciju u kojoj ima minimalnu visinu ili širinu ovisno o dimenzijama ploče. U svakom koraku rotacije se računa širina i visina te ako je manja od najmanje do sada pamte se njihovi iznosi kutova na kojima su postignuti.



Slika 3.2.2.1. Rotacija prekinuta sječenjem oblika.

Kada želimo rotirati neki oblik to radimo u koracima rotacije koji smo odredili u karakteristikama algoritma. Nakon svake rotacije se pregledava postoje li sječenja, ako postoje oblik se vraća jedan korak u natrag kao što je prikazano na slici 4.2.2.1. u kojoj je prikazana rotacija crvenog oblika u smjeru suprotnom od smjera kazaljke na satu, dok mu plavi oblik smeta. On se rotira do koraka u kojem se siječe s plavim

oblikom, nakon što se utvrdi sječenje plavi oblik se rotira za jedan korak natrag i završava se rotacija. Žuti oblik prikazuje položaj u kojemu će biti plavi oblik nakon rotacije. Isti postupak bi se dogodio i kada bi rotacija uzrokovala izlazak iz granica ploče. Analogno vrijedi i za rotaciju u smjeru kazaljke na satu.



Slika 3.2.2.2. Rotacija ne smeta.

Ako nema oblika s kojim bi se crveni oblik sjekao ili bi izašao iz ploče on će napraviti rotaciju za više od 360 stupnjeva. Kada ukupna rotacija postane veća od 360 stupnjeva oblik se rotira na početnu vrijednost, rotacija na početnu vrijednost se radi zbog mogućnosti odabira kuta rotacije koji se neće postići točno jedan cijeli krug, zatim se rotira na poziciju u kojoj zauzima najmanje visine ili širine, ovisno o tome koja je dimenzija ploče manja. To je omogućeno time što se nakon svakog koraka rotacije gleda koliko je širina i visina oblika a ako trenutna visina ili širina budu minimalne pamte se njihovi iznosi i kutovi rotacija na kojima su postignuti.

3.2.3. Kratke translacije

1. QTD (quick translate down) kratka translacija dolje. Pokušava pomaknuti ulazni oblik dolje za iznos njegove visine pomnožen s postotkom kratke translacije. Ako se nakon translacije oblik siječe s nekim drugim oblikom, onda se ulazni oblik vraća na početno mjesto.
2. QTL (quick translate left) kratka translacija lijevo. Pokušava pomaknuti ulazni oblik lijevo za iznos njegove širine pomnožen s postotkom kratke translacije. Ako se nakon translacije oblik siječe s nekim drugim oblikom, onda se ulazni oblik vraća na početno mjesto.
3. QTR (quick translate right) kratka translacija desno. Pokušava pomaknuti ulazni oblik desno za iznos njegove širine pomnožen s postotkom kratke translacije. Ako se nakon translacije oblik siječe s nekim drugim oblikom, onda se ulazni oblik vraća na početno mjesto.
4. QTU (quick translate up) kratka translacija gore. Pokušava pomaknuti ulazni oblik gore za iznos njegove širine pomnožen s postotkom kratke translacije. Ako se nakon translacije oblik siječe s nekim drugim oblikom, onda se ulazni oblik vraća na početno mjesto.

3.2.4. Trivijalne transformacije

Trivijalna transformacija u nekom smjeru (npr. dolje) gleda minimalnu Y koordinatu oblika i slobodno mjesta dolje. Oblik se uvijek pomiče za pola razlike između minimalnog Y i slobodnog mjesta. Nakon pomaka se provjerava siječe li se oblik s nekim drugim oblikom; ako ne onda se postupak ponavlja, a ako se siječe slobodno mjesto se postavi na polovicu udaljenosti između prethodnog slobodnog mjesta i minimalne Y koordinate, te se oblik vrati natrag na staru poziciju. Postupak se ponavlja dok razlika između minimalne Y koordinate i slobodnog mjesta ne postane manja od 1mm. Analogno vrijedi i za ostale translacije.

3.2.5. Nezavršni elementi

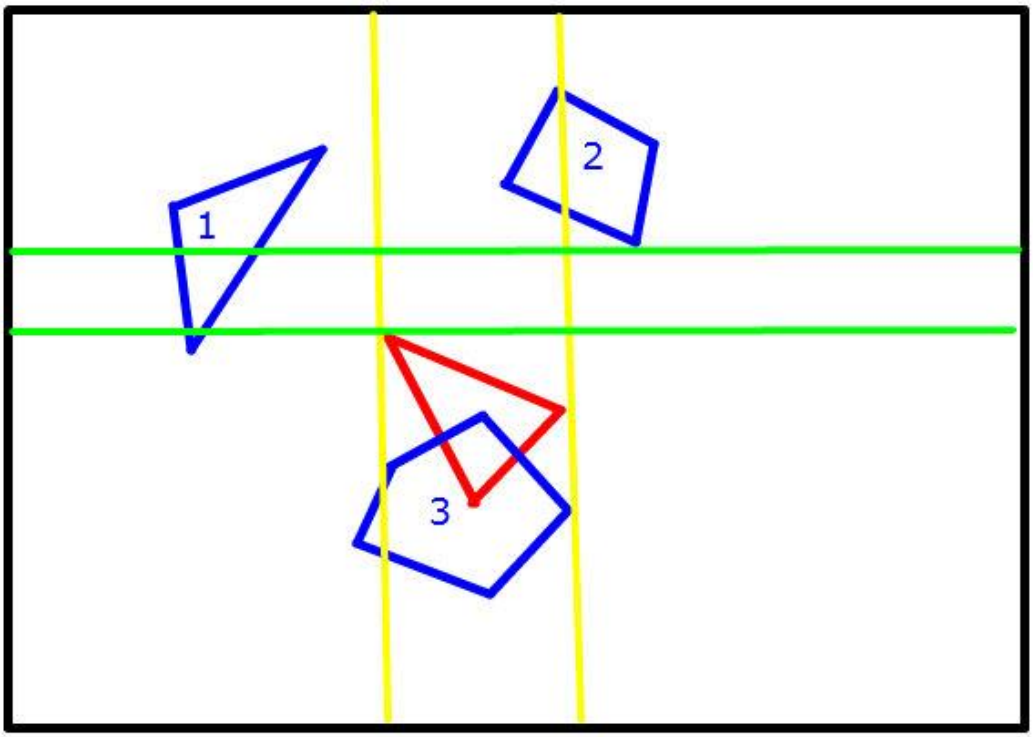
Popis nezavršnih elemenata :

1. IMSD (eng: Is more space down) ima li više mjesta dolje. On će izvršiti prvo dijete za zadani oblik i izračunati koliko je mjesta ostalo ispod njega, zatim će na ulaznim podatku (ne onome na kojemu se izvršilo prvo dijete) izvršiti drugo dijete te će onda na njemu izračunati koliko ima mjesta ispod toga oblika. Ako ima više mjesta ispod tog oblika nakon prvog djeteta izvršiti će se treće dijete a ako je više nakon drugog djeteta izvršiti će se četvrto dijete.

2. IMSL (eng: Is more space left) ima li više mjesta lijevo. Radi analogno postupku ispitivanja ima li više mjesta dolje samo su zamijenjene X i Y koordinatne osi.

3. IMSU (eng: Is more space up) ima li više mjesta gore. Radi analogno postupku ispitivanja ima li više mjesta dolje, s razlikom da se umjesto udaljenosti minimalne koordinate oblika od nule, gleda udaljenost maksimalne koordinate oblika od vrha ploče.

4. IMSR (eng: Is more space right) ima li više mjesta desno. Radi analogno postupku ispitivanja ima li više mjesta dolje ali sa svim razlika navedenim u postupku traženja mjesta lijevo i gore.



Slika 3.2.5.1. Logika nezavršnih elemenata.

Na slici vidimo crveni element i želimo provjeriti koliko ima mjesta iznad njega. Pretpostavimo da je ovo ploča koju smo dobili nakon izvršavanja prvog djeteta. Prvo postavimo da je slobodnog mjesto jednako udaljenosti vrha crvenog oblika i gornjeg kraja ploče. Zatim gledamo sve ostale elemente. Prvi plavi element se ne nalazi između žutih linja koje označavaju X koordinate crvenog oblika, on ne smeta i još uvijek možemo pomaknuti crveni oblik do kraja. Drugi plavi oblik smeta i smanjimo udaljenost za koju ga možemo pomaknuti na razliku minimalne Y koordinate plavog oblika broj 2 i maksimalne Y koordinate crvenog elementa što je prikazano zelenim linijama. Lako se može uočiti da bi se crveni element mogao pomaknuti za veću udaljenost, ali ovo je maksimalna udaljenost za koju smo sigurni da možemo pomaknuti crveni oblik. Za daljnje pomicanje su zadužene trivijalne transformacije. Na kraju gledamo za treći plavi obilk i vidimo da se on sječe s crvenim oblikom ali je ispod njega pa ga gledamo kao da ne smeta jer ako se ne sječe onda se neće sječi ni nakon što crveni oblik pomakne još gore, a ako se sječe onda se nadamo da će se pomaknuti dovoljno da se više neće sječi. Kada dobijemo tu udaljenost ponovimo taj postupak za ploču koju dobijemo nakon izvršavanja drugog djeteta, a onda ovisno o tim udaljenostima određujemo koje ćemo dijete izvršiti.

3.2.6. Tipovi algoritma

Tip algoritma se određuje pomoću uređene trojke, npr. (2,2,1) gdje brojevi imaju sljedeće značenje:

1. Broj izvršavanja algoritma na svakom obliku prije nego što prijeđemo na slijedeći lik: služi za omogućavanje kompleksnih transformacija koje su linearna kombinacija osnovnih transformacija koje su implementirane.

2. Broj izvršavanja algoritma nad svim likovima prije nego što prijeđemo na trivijalne transformacije. Zbog činjenice da su ulazni i izlazni podaci u algoritam isti, a izlaz je samo bolje postavljen možemo ga opet poslati kroz algoritam kako bi ga još više poboljšali, jako usporava program

3. Broj izvršavanja trivijalnih transformacija: trivijalne transformacije se izvršavaju nakon što su sve ostale transformacije gotove. One se izvršavaju ovim redoslijedom: prvo rotacija u smjeru suprotnom kazaljci na satu, zatim dolje pa lijevo. Broj označava koliko će se puta izvršiti svaka transformacija. Npr. 2 postupak će biti: rotacija, dolje, lijevo, rotacija, dolje, lijevo.

Tip algoritma ima velik utjecaj na vrijeme izvršavanja programa. Ovisno o parametrima algoritma vrijeme izvršavanja na jednoj ploči se može dobiti po ovoj formuli:

$$VrijmeIzvršavanja = brOblika * \big((prvaKonstanta * drugaKonstanta) + trecaKonstanta\big)$$

4. Ispitivanje

4.1. Opis ispitivanja

Sva ispitivanja se izvode na istom skupu ulaznih podataka, jedine stvari koje se mijenjaju su karakteristike algoritma, poput redoslijeda izvršavanja pojedinog oblika, broja izvršavanja algoritma za svaki oblik te mogućnosti korištenja različitih završnih i nezavršnih elemenata stabla, te veličini populacije i faktoru mutacije. Prvo ispitivanje obavljamo na dovoljno velikoj populaciji od 100 jedinki. Moguće je imati malo više od $1024 * 10^{16}$ različitih jedinki ako se koriste svi implementirani čvorovi stabla s minimalno dubinom stabla od 1 i maksimalnom dubinom stabla od 2. Broj različitih jedinki možemo dobiti prema ovoj formuli, gdje je

br– funkcija s parametrima m, M, NT, T

m – minimalna dubina stabla

M – maksimalna dubina stabla

NT – broj nezavršnih elemenata stabla

T – broj završnih elemenata stabla

$$br = \sum_{n=0}^{m-1}(NT * 4^n) + \sum_{n=m}^{M-1}[NT * 4^n * br(0, M-n, NT, T) + T * 4^n] + T * 4^M$$

4.2. Rezultati učenja

Odlučili smo kao uvjet zaustavljana koristiti 100 generacija bez poboljšanja dobrote. Svi rezultati učenja se nalaze u tablici koja je zajedno sa njihovim zapisnicima priložena ovom radu. Ovdje će biti iznesena njihova statistika. Najbolje i najlošije jedinke su imale dobrote:

$$najlošijadobrota = 1047$$

$$najboljadobrota = 1091$$

$$AritmetičkaSredina = 1073.82$$

U statistici su sve dobrote normalizirane tako da je njihova aritmetička sredina jednaka nuli kako bi odmah mogli vidjeti koji su rezultati bolji a koji lošiji od prosjeka.

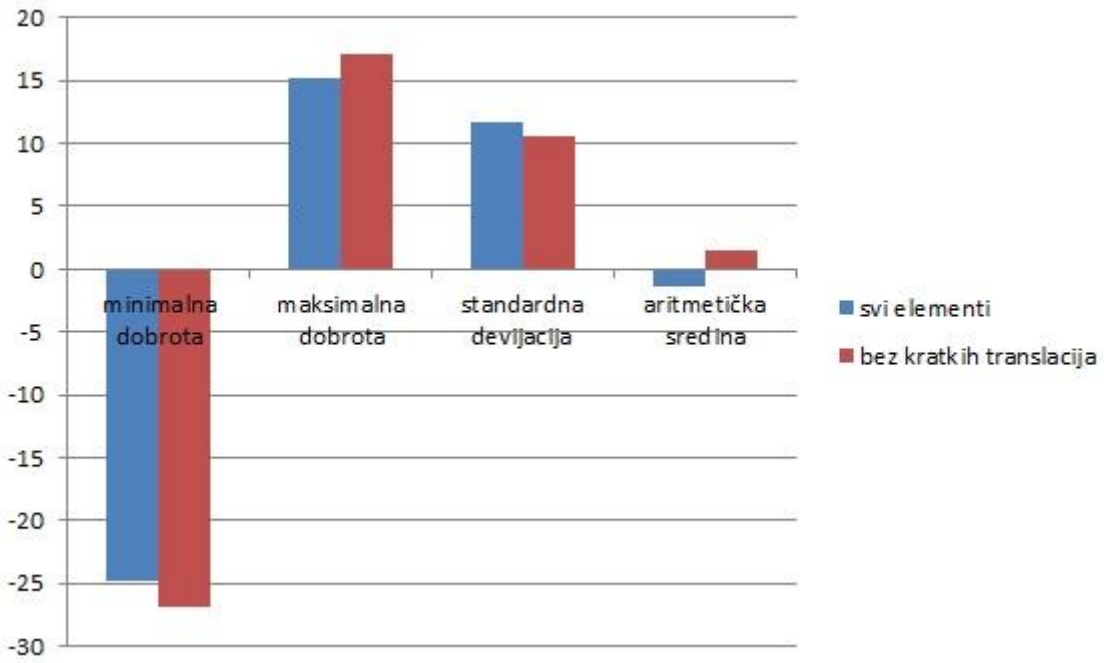
Detaljni rezultati ispitivanja mogu se vidjeti u poglavlju Dodaci na kraju ovog rada.

4.2.1. Ispitivanja o ovisnosti završnih elemenata

Prvo ćemo promatrati rješenja u odnosu na korištene završne elemente. Prvo promatramo rješenja u kojima je bila dopuštena uporaba svih završnih elemenata, a nakon toga one u kojima nije bilo dopušteno koristiti kratke translacije.

	svi dopušteni	bez kratkih translacija
minimalna dobrota	-24,86	-26,82
maksimalna dobrota	15,17	17,17
standardna devijacija	11,67	10,16
aritmetička sredina	-1,45	1,45

Tablica 4.2.1.1. Statistika ispitivanja ovisno o dopuštenim transformacijama.



Slika 4.2.1.1. Grafički prikaz statistike na ispitivanju različitog skupa završnih elemenata.

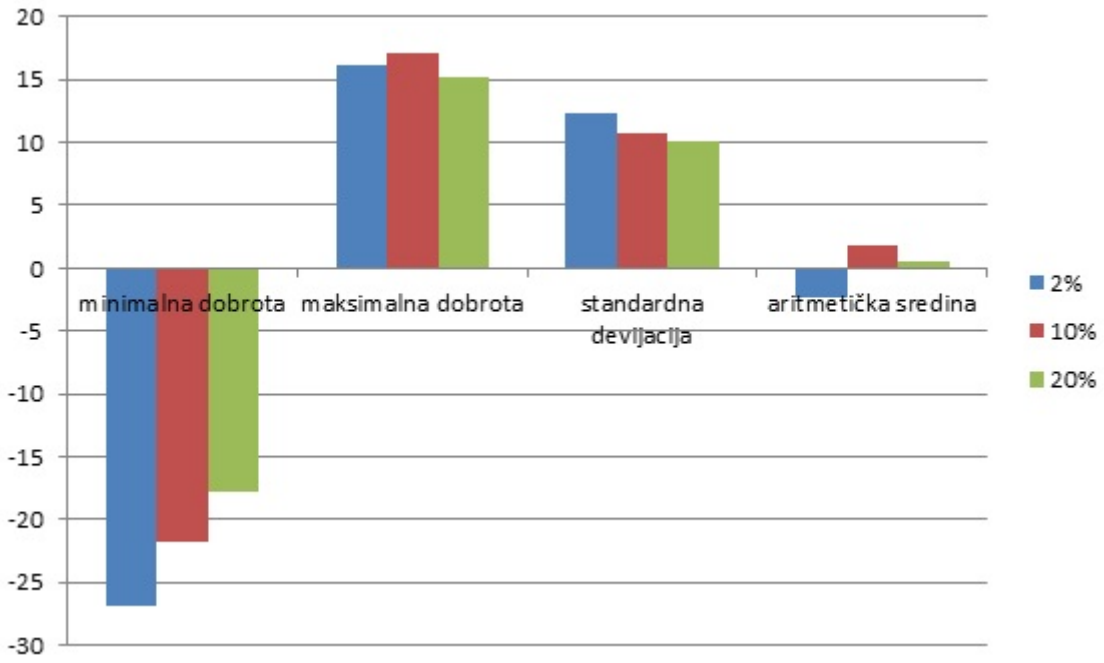
Ovdje vidimo da su najbolje i najlošije rješenje dosta blizu, a prosječna rješenje su bolja bez kratkih translacija i njihovo odstupanje od aritmetičke sredine je manje.

4.2.2. Ispitivanja o ovisnosti faktora mutacije

Sada ćemo usporediti skupove rješenja ovisno o faktoru mutacije, rješenja su podijeljena po faktorima mutacije od 2%, 10% i 20%. Opet gledamo kakvi će biti rezultati ako ovaj parametar bude stalan i za sve kombinacije ostalih parametara.

	2%	10%	20%
minimalna dobrota	-26,82	-21,82	-17,82
maksimalna dobrota	16,17	17,17	15,17
standardna devijacija	12,37	10,76	10,17
aritmetička sredina	-2,34	1,73	0,61

Tablica 4.2.2.1. Statistika ispitivanja ovisnosti o faktoru mutacije.



Slika 4.2.2.1. Grafički prikaz statistike ovisno o faktoru mutacije.

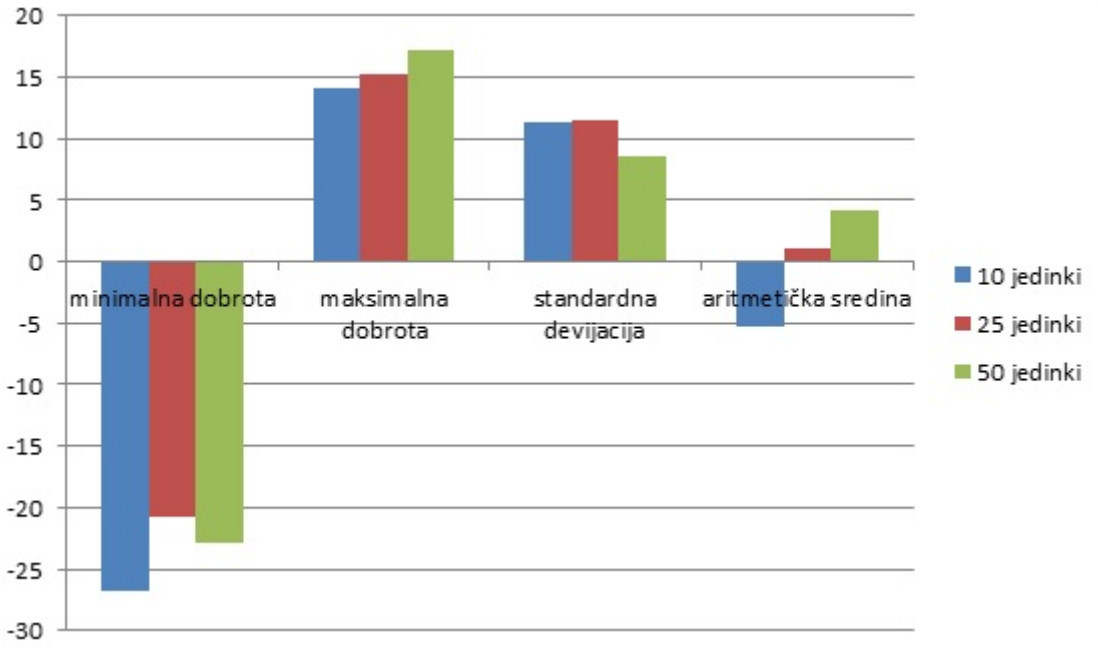
Ovdje vidimo da su rješenja s faktorom mutacije od 2% najlošija u svemu uključujući i najveću standardnu devijacija. Rješenja s faktorom mutacije od 10% imaju bolju prosječnu i maksimalnu dobrotu iako su najslabije dobrote puno bolje u skupu rješenja s mutacijom od 20%. Iz ovoga možemo zaključiti da je faktor mutacije od 20% dovoljno velik da sigurno izvuče rješenje iz lokalnih minimuma ali dosta sputava najbolje jedinke s slučajnim promjenama.

4.2.3. Ispitivanje o ovisnosti veličine populacije

Pogledajmo sada razlike u skupovima rješenja ovisno o veličini populacije, korištene populacije su od 10, 25 i 50 jedinki.

	10 jedinki	25 jedinki	50 jedinki
minimalna dobrota	-26,86	-20,82	-22,82
maksimalna dobrota	14,17	15,17	17,17
standardna devijacija	11,36	11,45	8,63
aritmetička sredina	-5,24	1,06	4,15

Tablica 4.2.3.1. Statistika ispitivanja ovisnosti o veličini populacije.



Slika 4.2.3.1. Grafički prikaz statistike ovisno o veličini populacije.

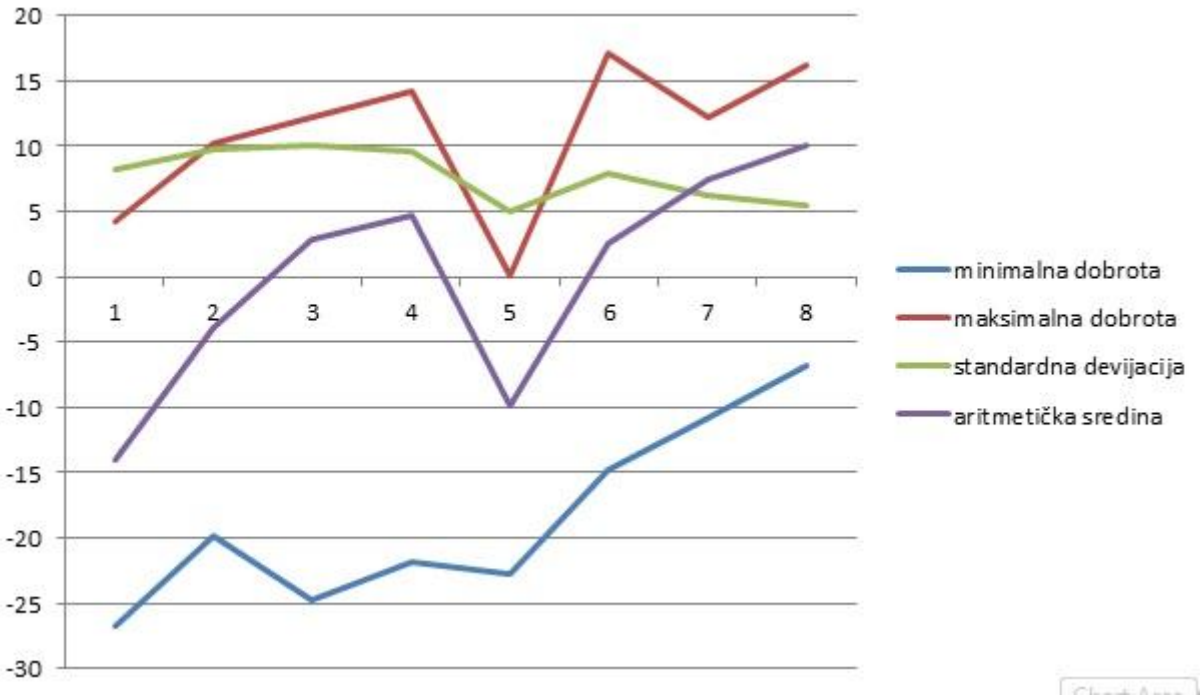
Ovdje vidimo da je najmanja populacija najlošija u svemu osim standardnoj devijaciji, što nije baš pohvalno je kaže da su sva rješenja približno jednako loša. Ispitivanje s populacijom od 50 jedinki ima jako veliku prosječnu vrijednost, i malu standardnu devijaciju iz čega se može zaključiti da neovisno o drugim parametrima rješenja uvijek uspijevaju biti puno bolja od prosjeka i malo odstupaju jedna od drugih.

4.2.4. Ispitivanje o ovisnosti tipa algoritma

Zadnja podjela rješenja je prema redoslijedu i broju prolaska kroz algoritam. Tipovi algoritma su detaljno objašnjeni u poglavlju 3.2.6. Ovdje ćemo ispitati tipove algoritama koje dobijemo svim kombinacijama brojeva 1 i 2.

	tip1 (111)	tip2 (211)	tip3 (121)	tip4 (221)	tip5 (112)	tip6 (212)	tip7 (122)	tip8 (222)
minimalna dobrota	-26,82	-19,82	-24,82	-21,82	-22,82	-14,82	-10,82	-6,82
maksimalna dobrota	4,17	10,17	12,17	14,17	0,17	17,17	12,17	16,17
standardna devijacija	8,16	9,83	10,07	9,59	4,97	7,91	6,23	5,48
aritmetička sredina	-13,99	-3,82	2,89	4,72	-9,93	2,61	7,39	10,11

Tablica 4.2.4.1. Statistika ispitivanja ovisnosti o tipu algoritma.



Slika 4.2.4.1. Graf usporedbe tipova algoritma.

Ovdje vidimo da tipovi 1, 2 i 5 imaju aritmetičku sredinu ispod nule što znači da su ispodprosječni. Najbolji tipovi su 7 i 8 dok je najbolje rješenje pripadalo tipu 6. Također vidimo da tipovi 7 i 8 imaju najmanju standardnu devijaciju što znači da neovisno o ostalim parametrima mogu dobivati jednako dobra rješenja, dok tip 5 neovisno o ostalim parametrima dobiva jednako loša rješenja.

4.3. Najbolje rješenje

```
484  Best fittnes : 1091,95199181084
485      TL
486      TD
487  IMSD
488          TL
489          TD
490      IMSD
491          TD
492          TU
493
494      RCCW
```

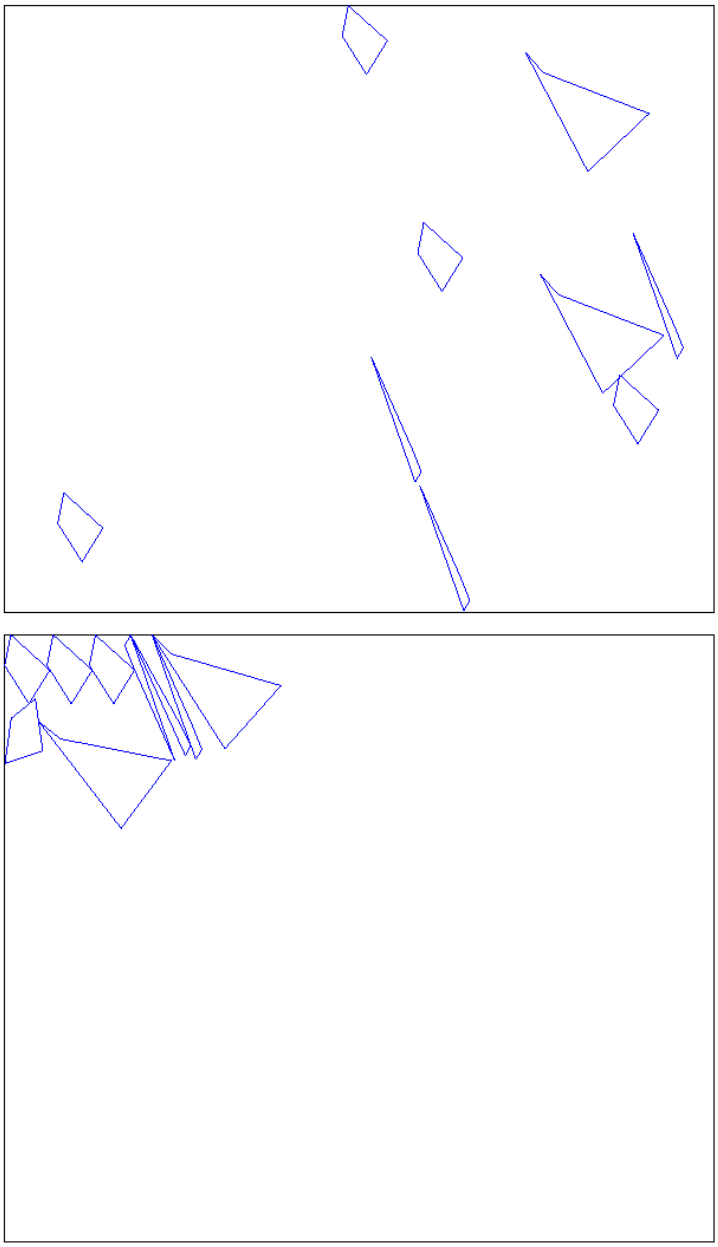
Slika 4.3.1. Najbolje rješenje.

Tip algoritma je 2, 1, 2. Ovo rješenje ćemo dalje ispitivati na ulaznim pločama koje ta jedinka još nije vidjela. Sada ćemo opisati kako radi ovo rješenje:

1. Pomakne oblik lijevo i desno na kopijama ploče, pa gleda kada ima više mjesta dolje.
2. Ako je više mjesta nakon pomaka lijevo idi na korak 3, a ako je više nakon pomaka dolje ide na korak 7.
3. Ponovno pomakne oblik lijevo i dolje na kopijama ploče, i opet gleda kada ima više mjesta dolje.
4. Ako je više mjesta nakon pomaka lijevo idi na korak 5, a ako je više nakon pomaka dolje idi na korak 6.
5. Pomakni oblik na pravoj ploči dolje, završi.
6. Pomakni oblik na pravoj ploči gore, završi.
7. Rotiraj oblik u smjeru suprotnom kazaljci na satu, završi.

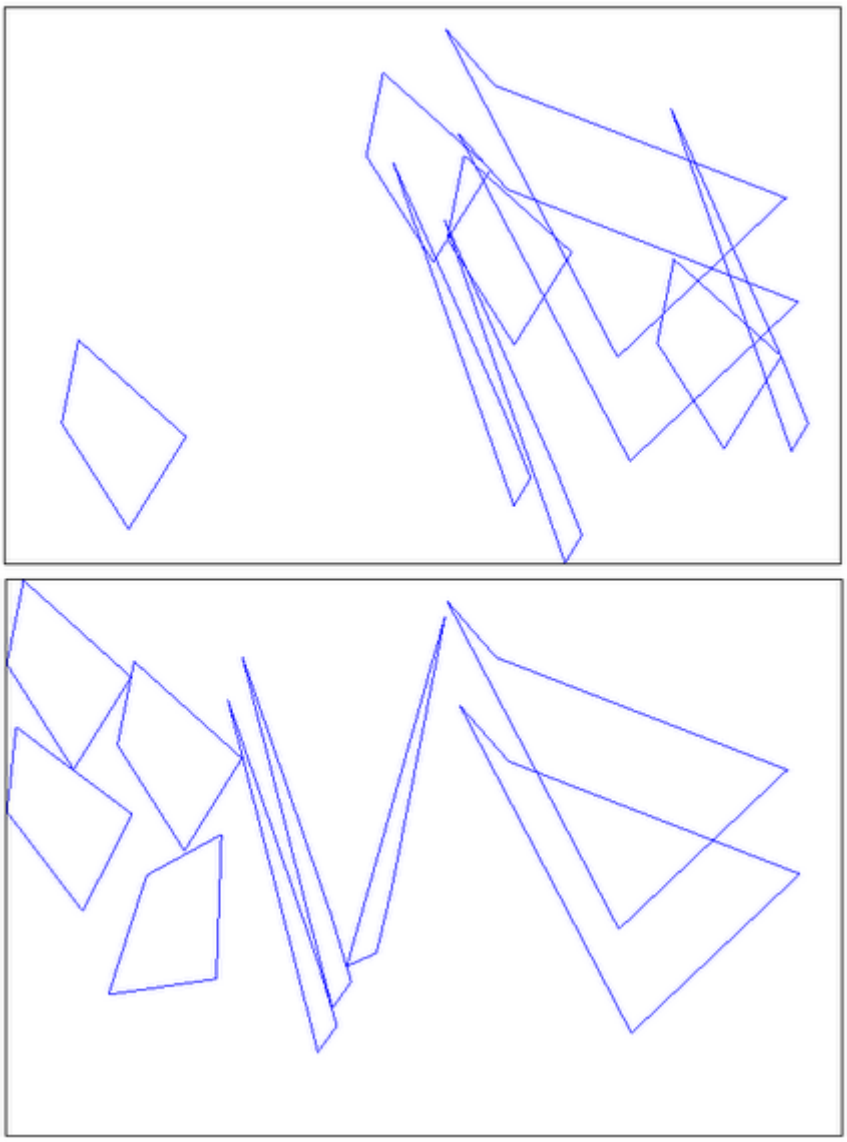
4.4. Rezultati ispitivanja

Sve slike su prikazane u početnom obliku, a odmah ispod nakon primjene algoritma.



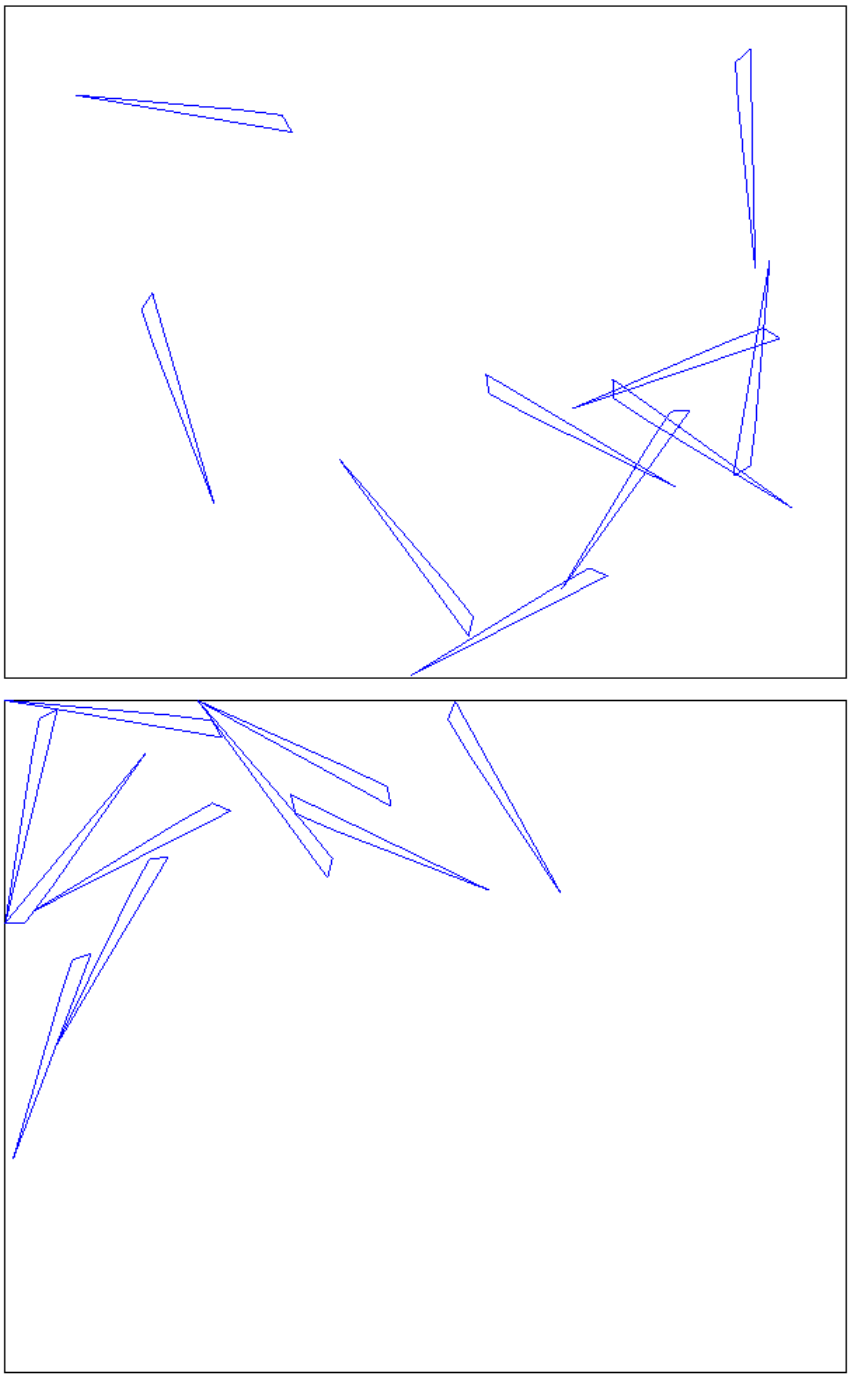
Slika 4.4.1. Ploča na kojoj je algoritam učio.

ovo je primjer ploče na kojoj je algoritam učio i to bi trebala biti jedna od najboljih ploča po tome kako će ih riješiti sve. Ovdje vidimo da je ploča puno veća od površine koje zatvaraju oblici pa ih ne bi trebalo biti teško posložiti da se ne sijeku.



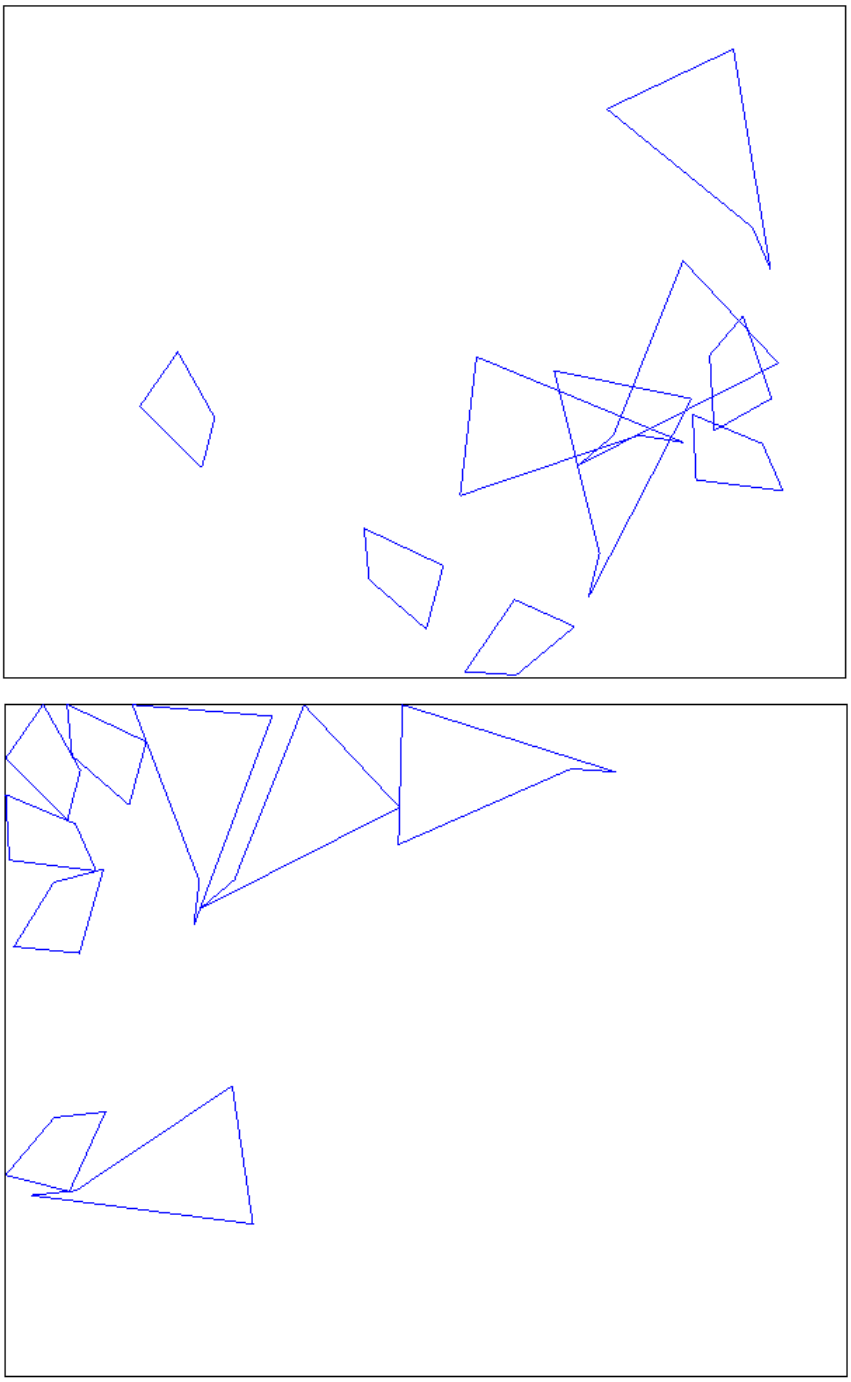
Slika 4.4.2. Smanjen dimenzije ploče.

Isti oblici kao na prethodnoj slici samo su dimenzije ploče smanjene. Vidimo da algoritam uspijeva puno poboljšati pozicije većine oblika iako ne uspijeva riješiti problem bez preklapanja.



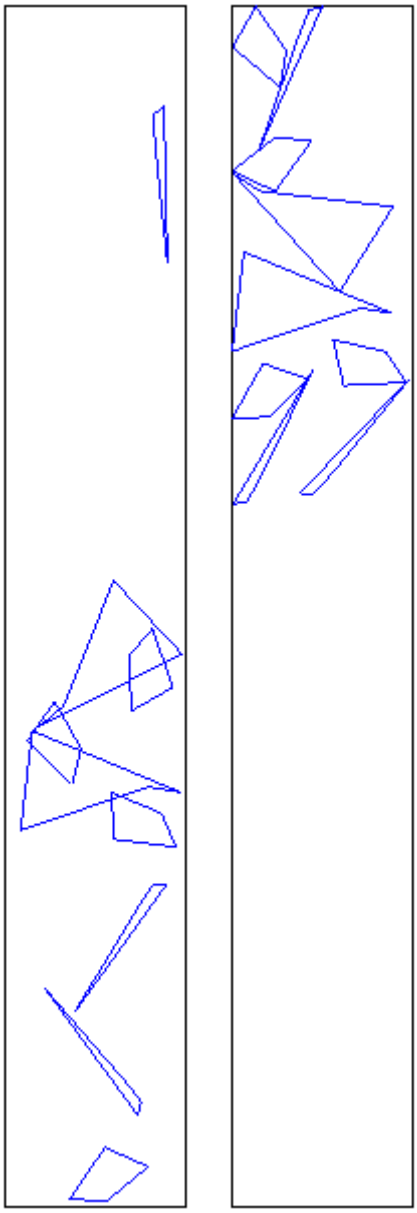
Slika 4.4.3. Samo duguljasti oblici.

Ovdje vidimo rezultate na primjeru s puno istih predmeta koji imaju jednu dimenziju puno veću od druge koji su najteži za dobro postaviti.



Slika 4.4.4. Samo oblici slični pravilnim mnogokutima.

Sličan primjer kao prethodni samo su oblici zamijenjeni za oblike koji su sličniji pravilnim mnogokutima. Rezultat je dosta bolji ali opet vidimo da bi nam trebao još jedan prolaz kroz trivijalne transformacije.



Slika 4.4.5. Jedna dimenzija ploče puno veća od druge.

Ovdje vidimo primjer s pločom koja ima jednu dimenziju puno manju od druge. Na ovoj slici se ploča prije algoritma nalazi na lijevoj strani a poslije na desnoj strani. Također opet vidimo malu rupu između oblika što bi značilo da bi opet morali povećati broj prolazaka kroz trivijalne transformacije.

5. Zaključak

U ovom radu rješavan je problem slaganja nepravilnih oblika na dvodimenzionalnu ploču. Problem je riješen uporabom genetskog programiranja, te kako je to pohlepan način rješavanja, rješenje nije najbolje ali je dovoljno dobro za upotrebu.

Genetskim programiranjem smo pokrili velik podskup mogućih algoritama te ih ispitali na primjerima koje oni nisu vidjeli tijekom učenja i dobili zadovoljavajuće rezultate s iznimkom većeg broja oblika na jednoj ploči. Taj problem možemo jednostavno riješiti povećavanjem broja trivijalnih transformacija .

Drugi nedostatak je možda loše definiran postupak rotacije u slučaju kada se dogodi preklapanje. Ovdje je oblik bio ostavljen u jednom koraku prije preklapanja, dok bi možda bilo poželjnije kao u slučaju kada nema preklapanja dodatno okrenuti oblik u položaj minimalne širine ili visine koju je postigao tijekom rotacije.

6. Literatura

[1] http://stackoverflow.com/questions/129389/how-do-you-do-a-deep-copy-an-object-in-net-c-specifically

[2] http://stackoverflow.com/questions/12251874/how-use-parallel-foreach-loop-instead-of-regular-foreach-info-about-parallel-f

[3] http://msdn.microsoft.com/en-us/library/system.drawing.drawing2d.aspx

[4] http://msdn.microsoft.com/en-us/library/system.threading.tasks.parallel.aspx

[5] http://en.wikipedia.org/wiki/Genetic_programming

[6] http://support.pixlr.com/pixlr/topics/how_to_make_straight_lines

[7] Pravac i ravnina; N. Elezović; FER

[8] Interaktivna računalna grafika; Marko Čupić i Željka Mihajlović; FER

[9] PrirodomInspiriraniOptimalizacijskiAlgoritmi; Marko Čupić; http://www.fer.unizg.hr/_download/repository/UI_13_PrirodomInspiriraniOptimizacijskiAlgoritmi.pdf ; FER 2012

7. Sažetak

7.1. Hrvatski

U ovom radu napravljen je opis problema slagana nepravilnih oblika na dvodimenzionalnu ploču. Taj problem je riješen uporabom genetskog programiranja i prikazani su rezultati.

Ključne riječi: genetsko, programiranje, nepravilni oblici, slaganje

7.2. English

In this work we have given description of problem of positioning of un-regular shapes on two-dimensional sheet. That problem is solved by use of genetic programing and we have shown results.

Key words: genetic, programing, irregular shapes, positioning

8. Dodaci

8.1. Tablica ispitivanja ovisnosti o dostupnosti završnih elemenata

kratkeTranslacije	faktorMutacije	velicinaPopulacije	brojJednogZaRedom	brojProlaskaKrozListu	brojTrivijalni	best result
da	2	10	1	1	1	1054
da	10	10	1	1	1	1054
da	20	10	1	1	1	1069
da	2	25	1	1	1	1053
da	10	25	1	1	1	1053
da	20	25	1	1	1	1056
da	2	50	1	1	1	1078
da	10	50	1	1	1	1071
da	20	50	1	1	1	1058
da	2	10	2	1	1	1057
da	10	10	2	1	1	1069
da	20	10	2	1	1	1056
da	2	25	2	1	1	1055
da	10	25	2	1	1	1077
da	20	25	2	1	1	1057
da	2	50	2	1	1	1082
da	10	50	2	1	1	1084
da	20	50	2	1	1	1077
da	2	10	1	2	1	1049
da	10	10	1	2	1	1086
da	20	10	1	2	1	1056
da	2	25	1	2	1	1086
da	10	25	1	2	1	1072
da	20	25	1	2	1	1086
da	2	50	1	2	1	1081
da	10	50	1	2	1	1082
da	20	50	1	2	1	1080
da	2	10	2	2	1	1056
da	10	10	2	2	1	1052
da	20	10	2	2	1	1076
da	2	25	2	2	1	1086
da	10	25	2	2	1	1081
da	20	25	2	2	1	1076
da	2	50	2	2	1	1084
da	10	50	2	2	1	1086
da	20	50	2	2	1	1083
da	2	10	1	1	2	1059

da	10	10	1	1	2	1061
da	20	10	1	1	2	1065
da	2	25	1	1	2	1059
da	10	25	1	1	2	1068
da	20	25	1	1	2	1065
da	2	50	1	1	2	1074
da	10	50	1	1	2	1064
da	20	50	1	1	2	1062
da	2	10	2	1	2	1059
da	10	10	2	1	2	1068
da	20	10	2	1	2	1074
da	2	25	2	1	2	1068
da	10	25	2	1	2	1081
da	20	25	2	1	2	1080
da	2	50	2	1	2	1068
da	10	50	2	1	2	1081
da	20	50	2	1	2	1079
da	2	10	1	2	2	1063
da	10	10	1	2	2	1086
da	20	10	1	2	2	1086
da	2	25	1	2	2	1076
da	10	25	1	2	2	1086
da	20	25	1	2	2	1086
da	2	50	1	2	2	1082
da	10	50	1	2	2	1081
da	20	50	1	2	2	1072
da	2	10	2	2	2	1067
da	10	10	2	2	2	1088
da	20	10	2	2	2	1085
da	2	25	2	2	2	1086
da	10	25	2	2	2	1089
da	20	25	2	2	2	1076
da	2	50	2	2	2	1082
da	10	50	2	2	2	1082
da	20	50	2	2	2	1085
ne	2	10	1	1	1	1047
ne	10	10	1	1	1	1056
ne	20	10	1	1	1	1056
ne	2	25	1	1	1	1057
ne	10	25	1	1	1	1056
ne	20	25	1	1	1	1056
ne	2	50	1	1	1	1071
ne	10	50	1	1	1	1061
ne	20	50	1	1	1	1071
ne	2	10	2	1	1	1054

ne	10	10	2	1	1	1072
ne	20	10	2	1	1	1074
ne	2	25	2	1	1	1072
ne	10	25	2	1	1	1074
ne	20	25	2	1	1	1069
ne	2	50	2	1	1	1069
ne	10	50	2	1	1	1080
ne	20	50	2	1	1	1082
ne	2	10	1	2	1	1076
ne	10	10	1	2	1	1071
ne	20	10	1	2	1	1070
ne	2	25	1	2	1	1086
ne	10	25	1	2	1	1086
ne	20	25	1	2	1	1076
ne	2	50	1	2	1	1077
ne	10	50	1	2	1	1083
ne	20	50	1	2	1	1078
ne	2	10	2	2	1	1076
ne	10	10	2	2	1	1076
ne	20	10	2	2	1	1076
ne	2	25	2	2	1	1082
ne	10	25	2	2	1	1080
ne	20	25	2	2	1	1086
ne	2	50	2	2	1	1086
ne	10	50	2	2	1	1088
ne	20	50	2	2	1	1084
ne	2	10	1	1	2	1065
ne	10	10	1	1	2	1061
ne	20	10	1	1	2	1061
ne	2	25	1	1	2	1065
ne	10	25	1	1	2	1065
ne	20	25	1	1	2	1065
ne	2	50	1	1	2	1051
ne	10	50	1	1	2	1069
ne	20	50	1	1	2	1071
ne	2	10	2	1	2	1064
ne	10	10	2	1	2	1075
ne	20	10	2	1	2	1080
ne	2	25	2	1	2	1086
ne	10	25	2	1	2	1081
ne	20	25	2	1	2	1079
ne	2	50	2	1	2	1081
ne	10	50	2	1	2	1091
ne	20	50	2	1	2	1081
ne	2	10	1	2	2	1075

ne	10	10	1	2	2	1082
ne	20	10	1	2	2	1085
ne	2	25	1	2	2	1086
ne	10	25	1	2	2	1086
ne	20	25	1	2	2	1086
ne	2	50	1	2	2	1081
ne	10	50	1	2	2	1077
ne	20	50	1	2	2	1086
ne	2	10	2	2	2	1085
ne	10	10	2	2	2	1081
ne	20	10	2	2	2	1079
ne	2	25	2	2	2	1085
ne	10	25	2	2	2	1086
ne	20	25	2	2	2	1088
ne	2	50	2	2	2	1090
ne	10	50	2	2	2	1088
ne	20	50	2	2	2	1089

8.2. Tablica ispitivanja ovisnosti o faktoru mutacije

faktorMutacije	kratkeTranslacije	velicinaPopulacije	brojJednogZaRedom	brojProlaskaKrozListu	brojTrivijalni	best result
2	da	10	1	1	1	1054
2	ne	10	1	1	1	1047
2	da	25	1	1	1	1053
2	ne	25	1	1	1	1057
2	da	50	1	1	1	1078
2	ne	50	1	1	1	1071
2	da	10	2	1	1	1057
2	ne	10	2	1	1	1054
2	da	25	2	1	1	1055
2	ne	25	2	1	1	1072
2	da	50	2	1	1	1082
2	ne	50	2	1	1	1069
2	da	10	1	2	1	1049
2	ne	10	1	2	1	1076
2	da	25	1	2	1	1086
2	ne	25	1	2	1	1086
2	da	50	1	2	1	1081
2	ne	50	1	2	1	1077
2	da	10	2	2	1	1056
2	ne	10	2	2	1	1076
2	da	25	2	2	1	1086
2	ne	25	2	2	1	1082
2	da	50	2	2	1	1084

2	ne	50	2	2	1	1086
2	da	10	1	1	2	1059
2	ne	10	1	1	2	1065
2	da	25	1	1	2	1059
2	ne	25	1	1	2	1065
2	da	50	1	1	2	1074
2	ne	50	1	1	2	1051
2	da	10	2	1	2	1059
2	ne	10	2	1	2	1064
2	da	25	2	1	2	1068
2	ne	25	2	1	2	1086
2	da	50	2	1	2	1068
2	ne	50	2	1	2	1081
2	da	10	1	2	2	1063
2	ne	10	1	2	2	1075
2	da	25	1	2	2	1076
2	ne	25	1	2	2	1086
2	da	50	1	2	2	1082
2	ne	50	1	2	2	1081
2	da	10	2	2	2	1067
2	ne	10	2	2	2	1085
2	da	25	2	2	2	1086
2	ne	25	2	2	2	1085
2	da	50	2	2	2	1082
2	ne	50	2	2	2	1090
10	da	10	1	1	1	1054
10	ne	10	1	1	1	1056
10	da	25	1	1	1	1053
10	ne	25	1	1	1	1056
10	da	50	1	1	1	1071
10	ne	50	1	1	1	1061
10	da	10	2	1	1	1069
10	ne	10	2	1	1	1072
10	da	25	2	1	1	1077
10	ne	25	2	1	1	1074
10	da	50	2	1	1	1084
10	ne	50	2	1	1	1080
10	da	10	1	2	1	1086
10	ne	10	1	2	1	1071
10	da	25	1	2	1	1072
10	ne	25	1	2	1	1086
10	da	50	1	2	1	1082
10	ne	50	1	2	1	1083
10	da	10	2	2	1	1052
10	ne	10	2	2	1	1076

10	da	25	2	2	1	1081
10	ne	25	2	2	1	1080
10	da	50	2	2	1	1086
10	ne	50	2	2	1	1088
10	da	10	1	1	2	1061
10	ne	10	1	1	2	1061
10	da	25	1	1	2	1068
10	ne	25	1	1	2	1065
10	da	50	1	1	2	1064
10	ne	50	1	1	2	1069
10	da	10	2	1	2	1068
10	ne	10	2	1	2	1075
10	da	25	2	1	2	1081
10	ne	25	2	1	2	1081
10	da	50	2	1	2	1081
10	ne	50	2	1	2	1091
10	da	10	1	2	2	1086
10	ne	10	1	2	2	1082
10	da	25	1	2	2	1086
10	ne	25	1	2	2	1086
10	da	50	1	2	2	1081
10	ne	50	1	2	2	1077
10	da	10	2	2	2	1088
10	ne	10	2	2	2	1081
10	da	25	2	2	2	1089
10	ne	25	2	2	2	1086
10	da	50	2	2	2	1082
10	ne	50	2	2	2	1088
20	da	10	1	1	1	1069
20	ne	10	1	1	1	1056
20	da	25	1	1	1	1056
20	ne	25	1	1	1	1056
20	da	50	1	1	1	1058
20	ne	50	1	1	1	1071
20	da	10	2	1	1	1056
20	ne	10	2	1	1	1074
20	da	25	2	1	1	1057
20	ne	25	2	1	1	1069
20	da	50	2	1	1	1077
20	ne	50	2	1	1	1082
20	da	10	1	2	1	1056
20	ne	10	1	2	1	1070
20	da	25	1	2	1	1086
20	ne	25	1	2	1	1076
20	da	50	1	2	1	1080

20	ne	50	1	2	1	1078
20	da	10	2	2	1	1076
20	ne	10	2	2	1	1076
20	da	25	2	2	1	1076
20	ne	25	2	2	1	1086
20	da	50	2	2	1	1083
20	ne	50	2	2	1	1084
20	da	10	1	1	2	1065
20	ne	10	1	1	2	1061
20	da	25	1	1	2	1065
20	ne	25	1	1	2	1065
20	da	50	1	1	2	1062
20	ne	50	1	1	2	1071
20	da	10	2	1	2	1074
20	ne	10	2	1	2	1080
20	da	25	2	1	2	1080
20	ne	25	2	1	2	1079
20	da	50	2	1	2	1079
20	ne	50	2	1	2	1081
20	da	10	1	2	2	1086
20	ne	10	1	2	2	1085
20	da	25	1	2	2	1086
20	ne	25	1	2	2	1086
20	da	50	1	2	2	1072
20	ne	50	1	2	2	1086
20	da	10	2	2	2	1085
20	ne	10	2	2	2	1079
20	da	25	2	2	2	1076
20	ne	25	2	2	2	1088
20	da	50	2	2	2	1085
20	ne	50	2	2	2	1089

8.3. Tablica ispitivanja ovisnosti o veličini populacije

velicinaPopulacije	kratkeTranslacije	faktorMutacije	brojJednogZaRedom	brojProlaskaKrozListu	brojTrivijalni	best result
10	da	2	1	1	1	1054
10	da	10	1	1	1	1054
10	da	20	1	1	1	1069
10	da	2	2	1	1	1057
10	da	10	2	1	1	1069
10	da	20	2	1	1	1056
10	da	2	1	2	1	1049
10	da	10	1	2	1	1086
10	da	20	1	2	1	1056

10	da	2	2	2	1	1056
10	da	10	2	2	1	1052
10	da	20	2	2	1	1076
10	da	2	1	1	2	1059
10	da	10	1	1	2	1061
10	da	20	1	1	2	1065
10	da	2	2	1	2	1059
10	da	10	2	1	2	1068
10	da	20	2	1	2	1074
10	da	2	1	2	2	1063
10	da	10	1	2	2	1086
10	da	20	1	2	2	1086
10	da	2	2	2	2	1067
10	da	10	2	2	2	1088
10	da	20	2	2	2	1085
10	ne	2	1	1	1	1047
10	ne	10	1	1	1	1056
10	ne	20	1	1	1	1056
10	ne	2	2	1	1	1054
10	ne	10	2	1	1	1072
10	ne	20	2	1	1	1074
10	ne	2	1	2	1	1076
10	ne	10	1	2	1	1071
10	ne	20	1	2	1	1070
10	ne	2	2	2	1	1076
10	ne	10	2	2	1	1076
10	ne	20	2	2	1	1076
10	ne	2	1	1	2	1065
10	ne	10	1	1	2	1061
10	ne	20	1	1	2	1061
10	ne	2	2	1	2	1064
10	ne	10	2	1	2	1075
10	ne	20	2	1	2	1080
10	ne	2	1	2	2	1075
10	ne	10	1	2	2	1082
10	ne	20	1	2	2	1085
10	ne	2	2	2	2	1085
10	ne	10	2	2	2	1081
10	ne	20	2	2	2	1079
25	da	2	1	1	1	1053
25	da	10	1	1	1	1053
25	da	20	1	1	1	1056
25	da	2	2	1	1	1055
25	da	10	2	1	1	1077
25	da	20	2	1	1	1057

25	da	2	1	2	1	1086
25	da	10	1	2	1	1072
25	da	20	1	2	1	1086
25	da	2	2	2	1	1086
25	da	10	2	2	1	1081
25	da	20	2	2	1	1076
25	da	2	1	1	2	1059
25	da	10	1	1	2	1068
25	da	20	1	1	2	1065
25	da	2	2	1	2	1068
25	da	10	2	1	2	1081
25	da	20	2	1	2	1080
25	da	2	1	2	2	1076
25	da	10	1	2	2	1086
25	da	20	1	2	2	1086
25	da	2	2	2	2	1086
25	da	10	2	2	2	1089
25	da	20	2	2	2	1076
25	ne	2	1	1	1	1057
25	ne	10	1	1	1	1056
25	ne	20	1	1	1	1056
25	ne	2	2	1	1	1072
25	ne	10	2	1	1	1074
25	ne	20	2	1	1	1069
25	ne	2	1	2	1	1086
25	ne	10	1	2	1	1086
25	ne	20	1	2	1	1076
25	ne	2	2	2	1	1082
25	ne	10	2	2	1	1080
25	ne	20	2	2	1	1086
25	ne	2	1	1	2	1065
25	ne	10	1	1	2	1065
25	ne	20	1	1	2	1065
25	ne	2	2	1	2	1086
25	ne	10	2	1	2	1081
25	ne	20	2	1	2	1079
25	ne	2	1	2	2	1086
25	ne	10	1	2	2	1086
25	ne	20	1	2	2	1086
25	ne	2	2	2	2	1085
25	ne	10	2	2	2	1086
25	ne	20	2	2	2	1088
50	da	2	1	1	1	1078
50	da	10	1	1	1	1071
50	da	20	1	1	1	1058

50	da	2	2	1	1	1082
50	da	10	2	1	1	1084
50	da	20	2	1	1	1077
50	da	2	1	2	1	1081
50	da	10	1	2	1	1082
50	da	20	1	2	1	1080
50	da	2	2	2	1	1084
50	da	10	2	2	1	1086
50	da	20	2	2	1	1083
50	da	2	1	1	2	1074
50	da	10	1	1	2	1064
50	da	20	1	1	2	1062
50	da	2	2	1	2	1068
50	da	10	2	1	2	1081
50	da	20	2	1	2	1079
50	da	2	1	2	2	1082
50	da	10	1	2	2	1081
50	da	20	1	2	2	1072
50	da	2	2	2	2	1082
50	da	10	2	2	2	1082
50	da	20	2	2	2	1085
50	ne	2	1	1	1	1071
50	ne	10	1	1	1	1061
50	ne	20	1	1	1	1071
50	ne	2	2	1	1	1069
50	ne	10	2	1	1	1080
50	ne	20	2	1	1	1082
50	ne	2	1	2	1	1077
50	ne	10	1	2	1	1083
50	ne	20	1	2	1	1078
50	ne	2	2	2	1	1086
50	ne	10	2	2	1	1088
50	ne	20	2	2	1	1084
50	ne	2	1	1	2	1051
50	ne	10	1	1	2	1069
50	ne	20	1	1	2	1071
50	ne	2	2	1	2	1081
50	ne	10	2	1	2	1091
50	ne	20	2	1	2	1081
50	ne	2	1	2	2	1081
50	ne	10	1	2	2	1077
50	ne	20	1	2	2	1086
50	ne	2	2	2	2	1090
50	ne	10	2	2	2	1088
50	ne	20	2	2	2	1089

8.4. Tablica ispitivanja ovisnosti o tipu algoritma

brojJednogZaRedom	brojProlaskaKrozListu	brojTrivijalni	kratkeTranslacije	faktorMutacije	velicinaPopulacije	best result
1	1	1	da	2	10	1054
1	1	1	ne	2	10	1047
1	1	1	da	10	10	1054
1	1	1	ne	10	10	1056
1	1	1	da	20	10	1069
1	1	1	ne	20	10	1056
1	1	1	da	2	25	1053
1	1	1	ne	2	25	1057
1	1	1	da	10	25	1053
1	1	1	ne	10	25	1056
1	1	1	da	20	25	1056
1	1	1	ne	20	25	1056
1	1	1	da	2	50	1078
1	1	1	ne	2	50	1071
1	1	1	da	10	50	1071
1	1	1	ne	10	50	1061
1	1	1	da	20	50	1058
1	1	1	ne	20	50	1071
2	1	1	da	2	10	1057
2	1	1	ne	2	10	1054
2	1	1	da	10	10	1069
2	1	1	ne	10	10	1072
2	1	1	da	20	10	1056
2	1	1	ne	20	10	1074
2	1	1	da	2	25	1055
2	1	1	ne	2	25	1072
2	1	1	da	10	25	1077
2	1	1	ne	10	25	1074
2	1	1	da	20	25	1057
2	1	1	ne	20	25	1069
2	1	1	da	2	50	1082
2	1	1	ne	2	50	1069
2	1	1	da	10	50	1084
2	1	1	ne	10	50	1080
2	1	1	da	20	50	1077
2	1	1	ne	20	50	1082
1	2	1	da	2	10	1049
1	2	1	ne	2	10	1076
1	2	1	da	10	10	1086
1	2	1	ne	10	10	1071
1	2	1	da	20	10	1056

1	2	1	ne	20	10	1070
1	2	1	da	2	25	1086
1	2	1	ne	2	25	1086
1	2	1	da	10	25	1072
1	2	1	ne	10	25	1086
1	2	1	da	20	25	1086
1	2	1	ne	20	25	1076
1	2	1	da	2	50	1081
1	2	1	ne	2	50	1077
1	2	1	da	10	50	1082
1	2	1	ne	10	50	1083
1	2	1	da	20	50	1080
1	2	1	ne	20	50	1078
2	2	1	da	2	10	1056
2	2	1	ne	2	10	1076
2	2	1	da	10	10	1052
2	2	1	ne	10	10	1076
2	2	1	da	20	10	1076
2	2	1	ne	20	10	1076
2	2	1	da	2	25	1086
2	2	1	ne	2	25	1082
2	2	1	da	10	25	1081
2	2	1	ne	10	25	1080
2	2	1	da	20	25	1076
2	2	1	ne	20	25	1086
2	2	1	da	2	50	1084
2	2	1	ne	2	50	1086
2	2	1	da	10	50	1086
2	2	1	ne	10	50	1088
2	2	1	da	20	50	1083
2	2	1	ne	20	50	1084
1	1	2	da	2	10	1059
1	1	2	ne	2	10	1065
1	1	2	da	10	10	1061
1	1	2	ne	10	10	1061
1	1	2	da	20	10	1065
1	1	2	ne	20	10	1061
1	1	2	da	2	25	1059
1	1	2	ne	2	25	1065
1	1	2	da	10	25	1068
1	1	2	ne	10	25	1065
1	1	2	da	20	25	1065
1	1	2	ne	20	25	1065
1	1	2	da	2	50	1074
1	1	2	ne	2	50	1051

1	1	2	da	10	50	1064
1	1	2	ne	10	50	1069
1	1	2	da	20	50	1062
1	1	2	ne	20	50	1071
2	1	2	da	2	10	1059
2	1	2	ne	2	10	1064
2	1	2	da	10	10	1068
2	1	2	ne	10	10	1075
2	1	2	da	20	10	1074
2	1	2	ne	20	10	1080
2	1	2	da	2	25	1068
2	1	2	ne	2	25	1086
2	1	2	da	10	25	1081
2	1	2	ne	10	25	1081
2	1	2	da	20	25	1080
2	1	2	ne	20	25	1079
2	1	2	da	2	50	1068
2	1	2	ne	2	50	1081
2	1	2	da	10	50	1081
2	1	2	ne	10	50	1091
2	1	2	da	20	50	1079
2	1	2	ne	20	50	1081
1	2	2	da	2	10	1063
1	2	2	ne	2	10	1075
1	2	2	da	10	10	1086
1	2	2	ne	10	10	1082
1	2	2	da	20	10	1086
1	2	2	ne	20	10	1085
1	2	2	da	2	25	1076
1	2	2	ne	2	25	1086
1	2	2	da	10	25	1086
1	2	2	ne	10	25	1086
1	2	2	da	20	25	1086
1	2	2	ne	20	25	1086
1	2	2	da	2	50	1082
1	2	2	ne	2	50	1081
1	2	2	da	10	50	1081
1	2	2	ne	10	50	1077
1	2	2	da	20	50	1072
1	2	2	ne	20	50	1086
2	2	2	da	2	10	1067
2	2	2	ne	2	10	1085
2	2	2	da	10	10	1088
2	2	2	ne	10	10	1081

2	2	2	da	20	10	1085
2	2	2	ne	20	10	1079
2	2	2	da	2	25	1086
2	2	2	ne	2	25	1085
2	2	2	da	10	25	1089
2	2	2	ne	10	25	1086
2	2	2	da	20	25	1076
2	2	2	ne	20	25	1088
2	2	2	da	2	50	1082
2	2	2	ne	2	50	1090
2	2	2	da	10	50	1082
2	2	2	ne	10	50	1088
2	2	2	da	20	50	1085
2	2	2	ne	20	50	1089