SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Raspodjela pravokutnih objekata u dvije dimenzije

Robert Pofuk

Voditelj: Domagoj Jakobović

Zagreb, travanj, 2010

Sadržaj

1. Uvod

Genetski algoritmi (engl. *Genetic Algorithms)* su prilagodljivi heuristički algoritmi pretraživanja koji se temelje na ideji evolucije, prirodne selekcije i genetici. Osnovni koncept genetskih algoritama je simulirati procese evolucije u prirodnim sustavima, posebice principe koje postavio Charles Darwin o preživljavanju najjačih (engl. *survival of the fittest*).

Genetski algoritmi se koriste za rješavanje problema u kojima je potrebno pretražiti velik broj mogućnosti kako bi se pronašlo rješenje. Također, mnogi problemi zahtijevaju prilagodljivost kako bi mogli nastaviti rad u okruženju koje se mijenja. Neki problemi zahtijevaju od programa da budi inovativni, da stvore nešto sasvim novo. Genetski algoritmi se koriste i za probleme koje je teško riješiti ručno, kao na primjer stvaranje umjetne inteligencije.

Jedan primjer primjene genetskih algoritama je rješavanje problema raspodjele pravokutnih objekata u dvije dimenzije. Taj problem najčešće pojavljuje u industriji kod pakiranja robe za transport. Algoritam se temelji na manipulaciji slojevima koji su popunjeni objektima. Cilj algoritma je evolucijom kroz generacije stvoriti raspored objekata koji se može smjestiti u spremnik fiksne širine i najmanje moguće duljine.

2. Genetski algoritmi

Naši životi su dominirani našim genima. Svaku živi organizam sastoji se od stanica koje sadrže jedan ili više kromosoma. Kromosomi služe kao nacrt ili skup pravila prema kojem je organizam izgrađen. Svaki kromosom je podijeljen na gene u kojima je pohranjena jedna informacija o organizmu (npr. boja očiju), svaka ta informacija naziva se fenotip (engl. *phenotype)*. Evolucija se temelji na dva procesa: rekombinaciji i mutaciji.

Rekombinacija je proces u kojem iz gena dviju jedinki roditelja, uzevši polovicu od svake, kreiraju geni za novu jedinku. Ovim postupkom nova jedinka dobije neka od svojstava oba roditelja.

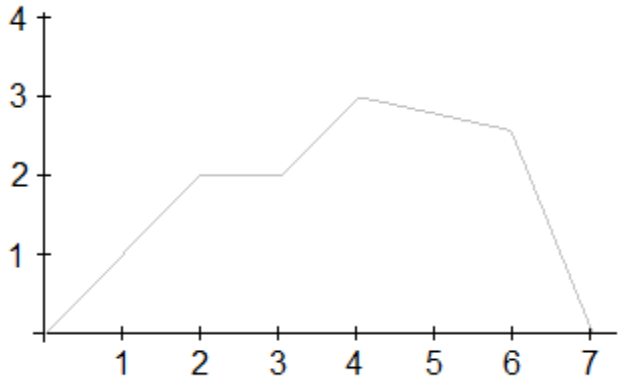
Mutacija je u prirodi spontan događaj u kojem se dio gena promijeni te se time promijeni i informaciju koja je u njemu pohranjena. Ovaj postupak za posljedicu ima nastanak jedinki sa nekim novim svojstvima koja toj jedinki daje prednost pred ostalima. Tako nastala jedinka stvara nove jedinke koje postaju dominantne zbog novog svojstva, odnosno omogućuje im da prežive kao najjače jedinke.

Kvaliteta gena koji su rezultat ovih procesa se određuje sposobnošću, odnosno vjerojatnošću, da će organizam preživjeti da se stvori potomke ili kao funkcija broja potomaka koje će taj organizam imati.

Razvoj evolucijskog računarstva počinje 1950. kada su znanstvenici počeli proučavati ideju evolucije kao alat za optimizaciju inženjerskih problema. Rochenberg je 1960. predstavio "evolucijske strategije" (engl. "evolution strategies"), metodu koju je upotrijebio za optimizaciju parametara aerodinamičnih profila. Fogel, Owens i Walsh su 1966. razvili "evolucijsko programiranje" (engl. "evolutionary programming") u kojem je rješenje problema bilo prikazano kao stroj s konačnim stanjima, koji je evoluirao nasumičnim mutiranjem prijelaza i odabirom najjačih.

Genetske algoritme je predložio John Holland 1970. Za razliku od evolucijske strategije i evolucijeskog programiranja, njegov cilj nije bio stvoriti algoritme za rješavanje određenog problema, već formalno proučavati fenomen prilagođavanja kakav pronalazimo u prirodi. (Mitchell, 1998) (Kim, 2001)

Osnovni elemente genetskih algoritama bit će opisani rješavajući problem pronalaženja maksimuma funkcije sa slike 2.1.



Slika 2.1: Funkcija

2.1. Prikaz jedinke

Prije nego se može riješiti problem genetskim algoritmom mora se pronaći način na koji će se kodirati problem. Za kodiranje kromosoma može se koristiti npr. niz realnih brojeva ili niz bitova kao na slici 2.2. Geni u kromosomu mogu biti predstavljeni kao pojedini bitovi ili kratki nizovi bitova.

1001 0110 1010 0011

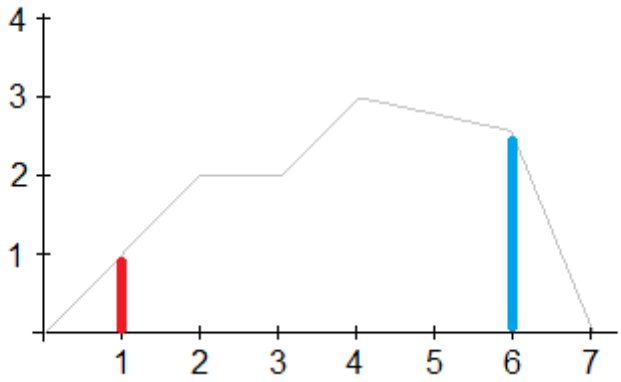
Slika 2.2: Kromosomi

U primjeru određivanja maksimuma funkcije, svaki bit predstavlja jedan gen u prikazu, a kromosom predstavlja binarni zapis broja u kojem se nalazi maksimum. Svaki kromosom sadrži četiri gena.

2.2. Inicijalizacija populacije

Najčešće se početna generacija stvara slučajnom odabirom vrijednosti gena, no postoje i mnoge druge metode, kao npr. uzimanje rješenja nekog drugog oblika optimizacije ili prethodnog izvođenja genetskog algoritma.

U primjeru funkcije stvore se 2 kromosoma (slika 2.3). Kromosom 1 ima binarnu vrijednost 1, a kromosom 2 vrijednost 6.



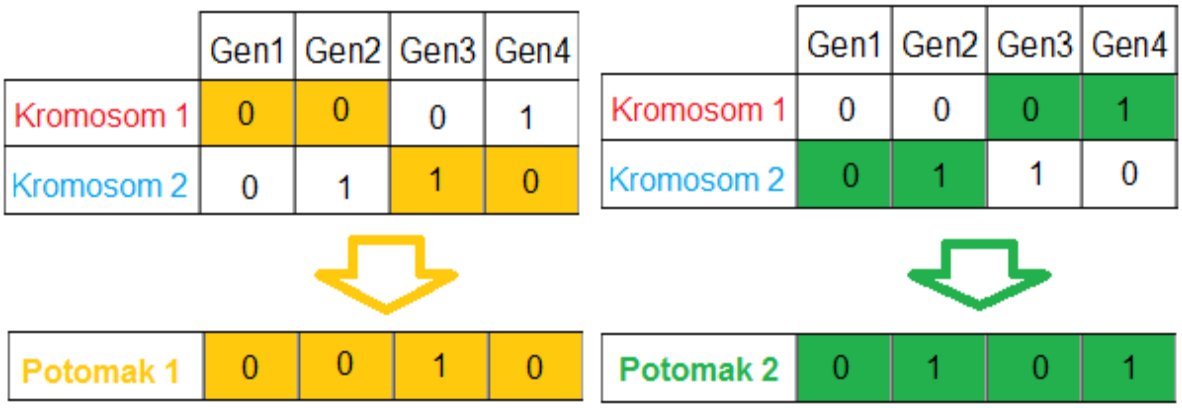
	Gen1	Gen2	Gen3	Gen4	Vrijednost funkcije
Kromosom 1	0	0	0	1	1,0
Kromosom 2	0	1	1	0	2,5

Slika 2.3: Početna populacija

Svaki kromosom predstavlja jednu jedinku populacije. Prilikom stvaranje populacije stvara se velik broj jedinki i svaka od njih predstavlja moguće rješenje problema.

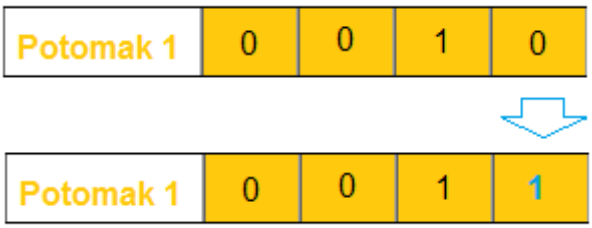
2.3. Genetski operatori

Genetski algoritam stvara nove jedinke sa novim kromosomima kombiniranjem postojećih u procesu koji se naziva križanje (eng. *crossover)* a dio je procesa rekombinacije. U križanju algoritam uzima dijelove kromosoma dviju postojećih jedinki i spaja ih u novi kromosom čime se stvara novo rješenje problema. Operacije križanja ovisi o prikazi kromosoma i u nekim situacijama može biti jako komplicirana. Samu operaciju križanja lako je implementirati, ali problem je prilagoditi ju za određeni problem kako bi se poboljšale performanse izvođenja. Primjer križanja prikazan je slikom 2.4.



Slika 2.4: Križanje

Nakon što izvede križanje, a prije završavanja stvaranja novih jedinki, algoritam izvodi operaciju mutacije (slika 2.5). Operacije mutacije izvodi slučajne male promjene unutar kromosoma. Isto kao i križanje, mutacija, ovisi o načinu kodiranja. Ona sprečava da sva rješenja upadnu u lokalni minimum i proširuje područje pretraživanja algoritma.

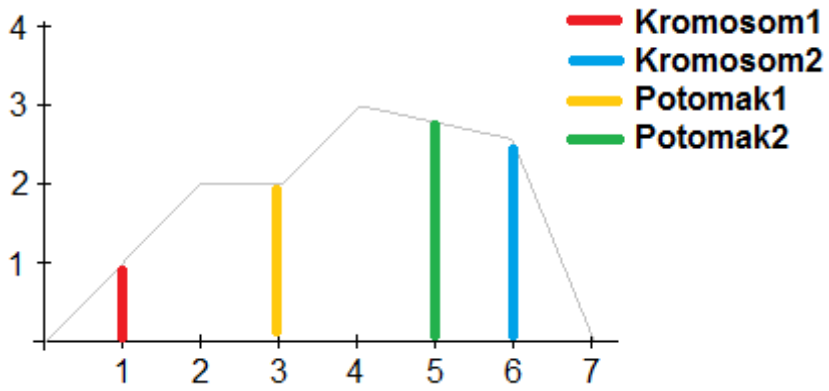


Slika 2.5: Mutacija

Operacije križanja i mutacija se ne izvode kod svakog stvaranja nove populacije. Dali će se križanje ili mutacija izvršiti ovisi o dva parametra genetskog algoritma koji se nazivaju:

- vjerojatnost križanja (engl. crossover probability*)*
- vjerojatnost mutacije (engl. *mutation probability)*

Vjerojatnost križanja je najčešće visoka i uzima se u intervalu 0.6 do 1.0, dok je vjerojatnost mutacije niska, najčešće manja od 0.1, ali kod nekih problema veća vjerojatnost mutacija daje bolje rezultate. Visoka vjerojatnost mutacije pretvara genetski algoritam u algoritam slučajnog pretraživanja. U slučaju kada se ne izvede nijedna operacija tada algoritam stvara novu generaciju kopiranjem roditelja. Za primjer funkcije, druga generacija kromosoma izgleda kao na slici 2.6.



Slika 2.6: Druga generacija

2.4. Funkcija dobrote

Nakon što se stvore nove jedinke, odnosno rješenja, posljednja operacija koju genetski algoritmi izvršavaju nad novom generacijom je funkcija dobrote. Funkcija dobrote procjenjuje kvalitetu (dobrotu) rješenja nastalih rekombinacijom i mutacijom. Funkcija dobrote je jedinstvena za svaki problem i zapravo govori algoritmu što dalje optimizirati. Usporedbom dobrote govori se algoritmu dali treba povećati ili smanjiti dobrotu kromosoma.

Izbor roditelja koji će poslužiti za stvaranje nove generacije se zove selekcija (engl. S*election).* Selekcija se najčešće vrši na temelju dobrote kromosima. Na ovaj način se odabiru kromosomi čijim će kombiniranjem nastati najbolji potomci. Isto takav proces odabira pronalazimo u prirodi. Od kromosoma koji imaju najveću faktor dobrote očekuje se da stvori najviše potomaka. Ovakav način odabira potomaka ima veliki nedostatak. Kao na primjeru funkcije, svi će kromosomi s vremenom početi izgledati jednako, što smanjuje područje istraživanja i što može uzrokovati da svi kromosomi teže prema lokalnom maksimumu. Kako bi se omogućilo da algoritam pronađe bolje rješenje često se odabiru slučajni roditelji.

Jedna od češće korištenih metoda odabira potomaka zove se "*Roulette Wheel Selection*" . Algoritam odabira prolazi kroz sljedeće korake:

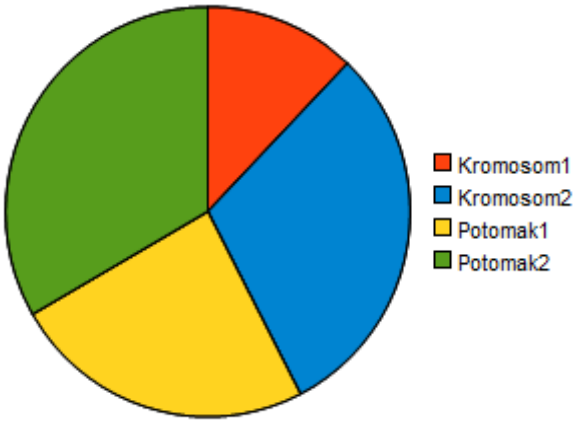
- računa se suma dobrote svi jedinki populacije, nazovimo ju ukupna dobrota F_{sum}
- odabire se slučajan broj (n) između 0 i F_{sum}
- uzima se prva jedinku iz populacije čija dobrota, pridodana dobrotama prethodnih jedinki, prelazi broj n

Za funkciju sa slike 2.1 imamo, nakon druge generacije vrijednosti u tablici. Iznos dobrote određujemo prema vrijednosti funkcije. Prikažemo tabelu u obliku kružnog grafa (slika 2.7). Podijelimo kružnicu na F_{sum} dijelova, te svaki dio nazovemo jednim brojem od 1 do F_{sum} . Potomak2 je najbolji i zauzima najveću površinu na grafu, dok kromosom1 zauzima, zbog najmanje dobrote, najmanju površinu.

Tablica 2.1: Dobrota jedinki

	Geni	Vrijednost funkcije	Dobrota
Kromosom1	**0001**	**1,00**	**100**
Kromosom2	**0110**	**2,50**	**250**
Potomak1	**0011**	**2,00**	**200**
Potomak2	**0101**	**2,75**	**275**

Treći korak selekcije govori da se uzima polje na grafu unutar kojeg se nalazi n - ti dio kružnice. (Kim, 2001)



Slika 2.7: Roulette Wheel Selection

Drugi način provođenja selekcije je turnirski odabir (engl. *Tournament Selection*). Turnirski odabir se temelji na provođenju turnira između određenog broja jedinki. Pobjednik turnira je jedinka s najvećom dobrotom. Prosjek dobrote na ovaj način odabranih jedinki je veći od ukupnog prosjeka svih jedinki. (Miller, 1995)

2.5. Uvjet prekida

Trajanje izvođenja genetskog algoritma nije ograničeno te se mora odrediti uvjet koji će ga prekinuti. Kao uvjet prekida može biti jedan od sljedećih:

- Broj generacija: algoritam se zaustavlja kad broj generacija dosegne određenu vrijednost
- Vremensko ograničenje
- Ograničenje dobrote: izvođenje algoritma se prekida kad dobrota najbolje jedinke dosegne određenu vrijednost
- Prekidanje algoritma kada kroz određen broj generacija nije bilo poboljšanja dobrote jedinki
- Prekidanje kada određeni vremenski period nije bilo poboljšanja dobrote jedinki

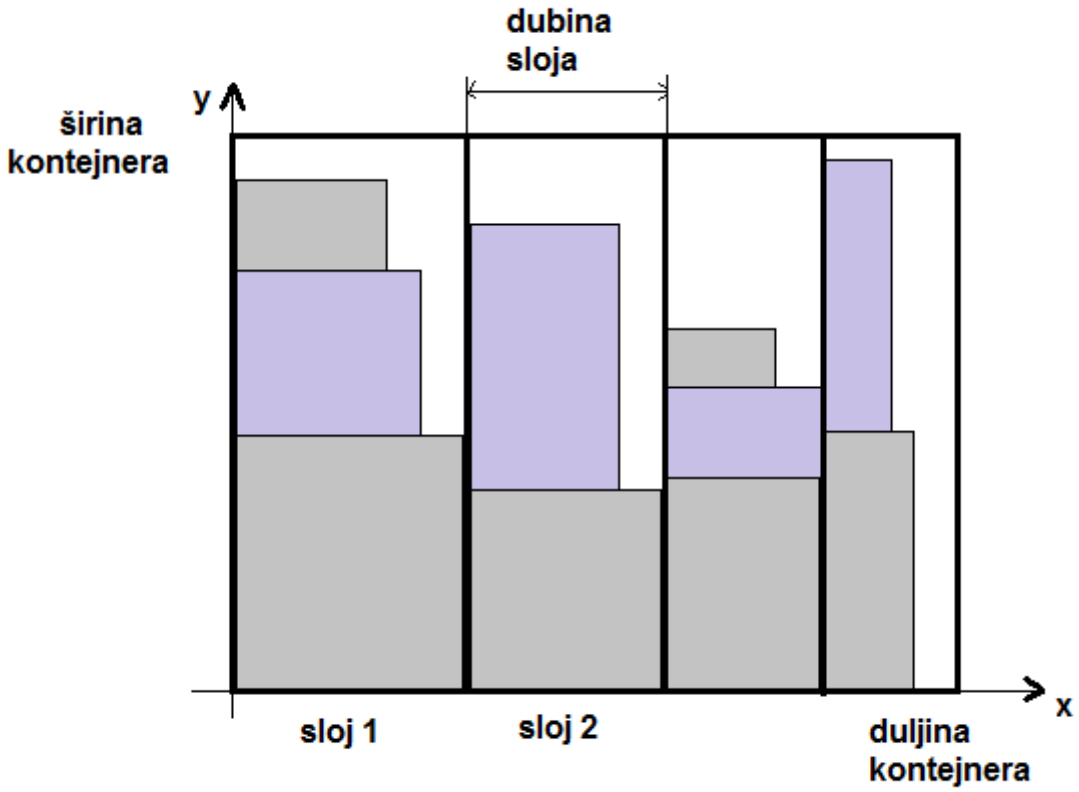
3. Primjena genetskog algoritma na problem raspodjele pravokutnih objekata u dvije dimenzije

Primjer primjene genetskih algoritama je problem smještaja pravokutnih objekata u spremnik pravokutnog oblika, fiksne širine i promjenjive dužine. Mnogi algoritmi za rješavanje problema koriste kodirana rješenja koja su kreirana korištenjem standardnih genetskih operacija. U rješenju koje ćemo opisati ne koristi se kodiranje rješenja već se genetski algoritmi koriste za manipulacijom potpuno definiranih slojeva. Cilj algoritma je smještanje oblika u spremnika najmanje duljine. Svaki sloj je prihvatljiv ako se oblici ne preklapaju. Svi oblici unutar spremnika su postavljeni paralelno sa stranicama spremnika.

3.1. Prikaz objekata i funkcija dobrote

Genetski algoritam generira raspored pomoću slojeva. Svaki raspored se sastoji od uzastopnih pravokutnih slojeva na koje su raspoređeni jedan ili više objekata s uvjetom da svaki objekt pripada samo jednom sloju. Širina svakog sloja jednaka je širini spremnika, dok je dubina određena sa najdužim objektom. Izgled rasporeda prikazan je na slici 3.1.

Traženje rješenja se izvršava u prostoru potpuno definiranih rasporeda sa slojevima. Struktura podataka koja se koristi za prikaz rasporeda prikazana je slikom 3.2.

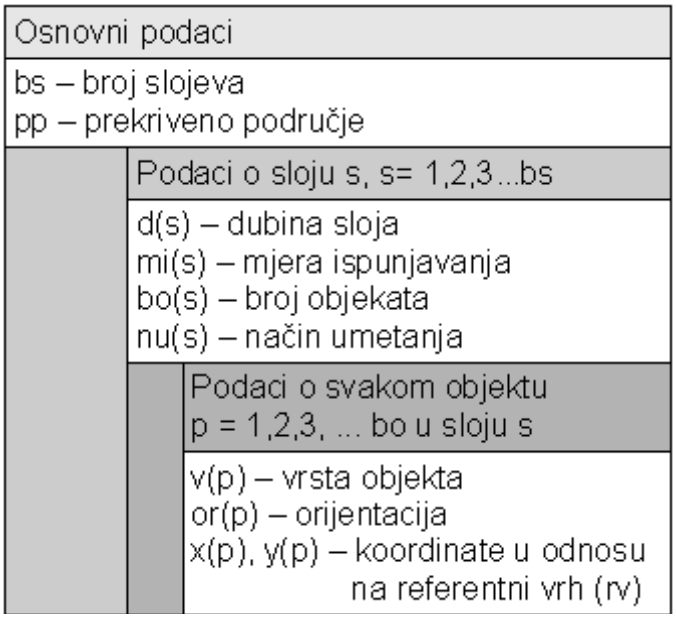


Slika 3.1: Slojevi

Osnovni elementi strukture podataka su sljedeći:

- vrsta objekta, *v(p)*, definirana je njegovom dimenzijom i oblikom, npr. jedan tip objekata predstavljaju sukladni pravokutnici. Tipovi su numerirani silazno prema

površini koju zauzimaju



Slika 3.2: Struktura podataka

- orijentacija objekta, *or(p)*, jednaka je 0 ako kraći rub objekta leži paralelno sa širinom spremnika odnosno 0 ako je suprotno

- referentni vrh, *rv(p)*, u prikazu je vrh objekta koji je najbliži ishodištu koordinatnog sustava

- vrsta *v(p)*, orijentacija or(p) i koordinate *x(p)* i *y(p)*, objekta *p*, predstavljaju poziciju objekta unutar sloja

- mjera ispunjavanja, *mi(s)*, definirana je kao kvocijent površine svih objekata unutar sloja i površine cijelog sloja

- način umetanja, *nu(s)* označava način na koji se stvaranju novi prazni prostori prilikom umetanja objekta

- slojevi su u prikazu uvijek raspoređeni silazno u odnosu na brzinu ispunjavanja

- vrijednost prekrivenoga dijela spremnika služi kao iznos dobrote određenog rješenja

Struktura podataka u potpunosti definira svaki prikaz pa nije potrebno dodatno kodiranje rješenja. (Andreas, 2003)

3.2. Inicijalizacija populacije

Za stvaranje početne generacije, koristi se BFDH algoritam (engl. *Best Fit Decreasing Height Heuristic).* Pseudokod algoritma dan je slikom 3.3. Na ovaj način dobijemo velik broj slojeva koji imaju veliku mjeru ispunjavanja. Također, izvođenjem ovog algoritma dobijemo početnu duljinu (*poc_duljina*) spremnika koji koristimo u izvođenju algoritma. Standardnom algoritmu možemo dodati još dvije mogućnosti:

```
int BFDH() {
     sve objekte okreni da je sirina veca ili jednaka duljini;
     sortiraj ih silazno po duljini;
     stvori_novi_sloj(sirina = sirina_kontejnera,
                      duljina = duljina_objekta);
     za (sve objekte) {
          ako(objekt stane u neki postojeci sloj)
               stavi ga u taj sloj;
          inace
               stvori_novi_sloj(duljin = duljina_objekta);
          }
     }
```

Slika 3.3: BHFD

- prilikom stavljanja svakog objekta omogućuje se promjena orijentacije (*or*)

- nakon što je neki sloj napunjen, sortiramo sve objekte prema duljini stranice paralelne sa duljinom spremnika. Dobiju se slobodna mjesta iznad svih objekata, traži se najveći objekt koji stane u jedno od novonastalih mjesta, te se smjesti u donji lijevi kraj. Ponovno se računaju slobodni prostori i traži se dali koji objekt stane u prostor. Ponavljamo sve dok ne postoji objekt koji stane u neki slobodni prostor sloja.

Pretraživanje genetskog algoritma ograničimo sa vrijednostima *max_objekata* i *min_objekata* na sljedeći način:

- stvorimo početno rješenje sa BHFD i sortiramo slojeve prema mjeri ispunjavanja

- uzimamo slojeve od najveće mjere ispunjavanje kao gotova rješenja sve dok broj objekata u preostalim slojevima nije između *max_objekata* i *min_objekata*

3.3. Genetski operatori

Genetski operatori izvode se na jednom ili više gena. Geni su u ovom slučaju slojevi. Svaki sloj predstavlja jedan gen. Broj jedinki u svakoj generaciji je ograničen na *broj_jedinki*. Na početku svake nove generacije izvode se sljedeći koraci:

- potomak je postavljen kao prazan (nema nijednog sloja), te su u njega preneseni neki najbolji slojevi iz prethodne generacije

- nakon prijenosa slojeva potomak se može nadopuniti sa slojevima

Prijenos slojeva ovisi o operaciji koja se izvodi.

3.3.1 Križanje

Križanje je jedan od načina prijenosa slojeva u potomka. Križanje se izvodi na najboljih *broj_krizanja* jedinki iz prijašnje generacije (engl. *Elitist strategy).* Prilikom odabira slojeva, najbolji slojevi od oba roditelja su odabrani gledajući mjeru ispunjavanja. Prvi sloj

od roditelja je uvijek prenesen dok prijenos ostalih ovisi o tome postoje li objekti koji ga ispunjavanju (dali nisu već neki od njih već iskorišteni) i da li dodavanjem sloja, duljina svih slojeva u potomku ne prelazi duljinu spremnika.

3.3.2 Mutacija

Postoje dvije vrste mutacija: klasična mutacija i granična mutacija. Prilikom klasične mutacije slojevi koji se prenose odabrani su slučajnim odabirom te u prijenosu sudjeluje najviše polovica od svih slojeva. Kod granične mutacije u potomka se prenose svi osim dva slučajno odabrana sloja.

3.3.3 Završavanje rješenja

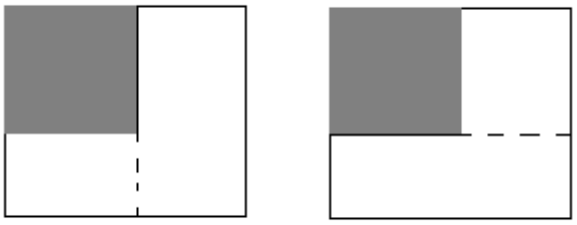
Nakon genetskih operacija treba u svakog potomka smjestiti preostale objekte na način da stvorimo nove slojeve. Novi sloj stvori se tako da prema slučajno odabranom objektu odredimo dubinu sloja, a zatim se taj sloj napuni sa slobodnim objektima.

3.3.4 Parametri i uvjet prekida

Parametrima genetskog algoritma može se značajno utjecati na kvalitetu rezultata. Algoritam raspoređivanja ima sljedeće parametre:

- širina spremnika
- broj jedinki u svakoj generaciji
- vjerojatnost križanja
- vjerojatnost mutacija = 1 – vjerojatnost križanja
- broj generacija
- rotacija
- način stvaranja slobodnih prostora (prazni)

Prilikom umetanja jednog objekta u neki prazni prostor nastaju dva nova slobodna prostora. Oni mogu nastati na dva načina prikazana slikom 3.4.

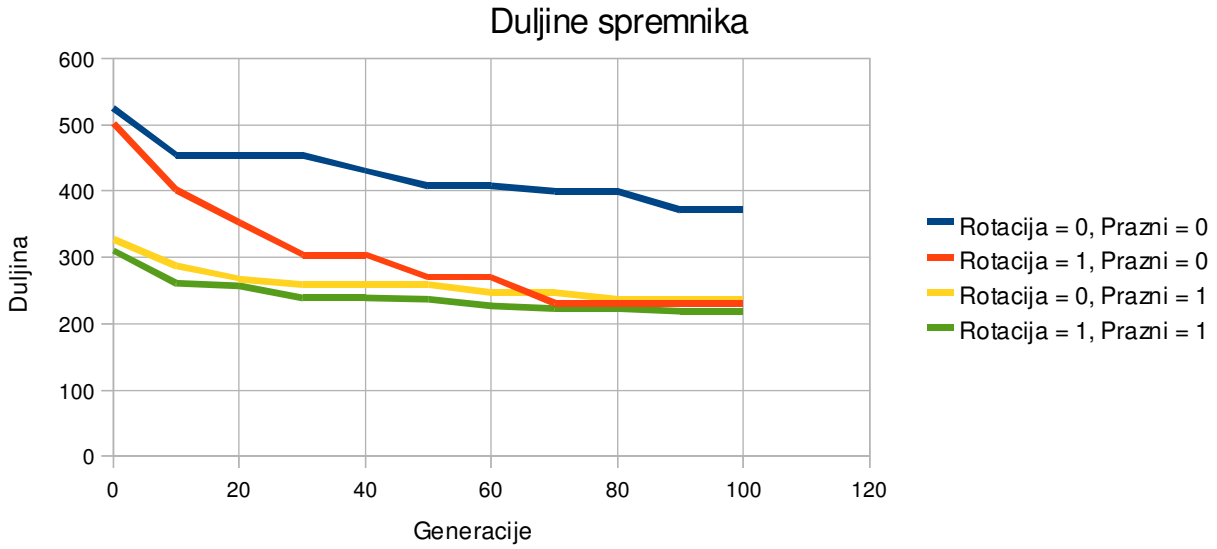


Slika 3.4: Stvaranje slobodnih prostora

Uvjet prekida algoritma je broj generacija. Algoritam se može također prekinuti nakon što nije nastala nijedna jedinka sa duljinom manjom od minimalne duljine iz prethodne generacije.

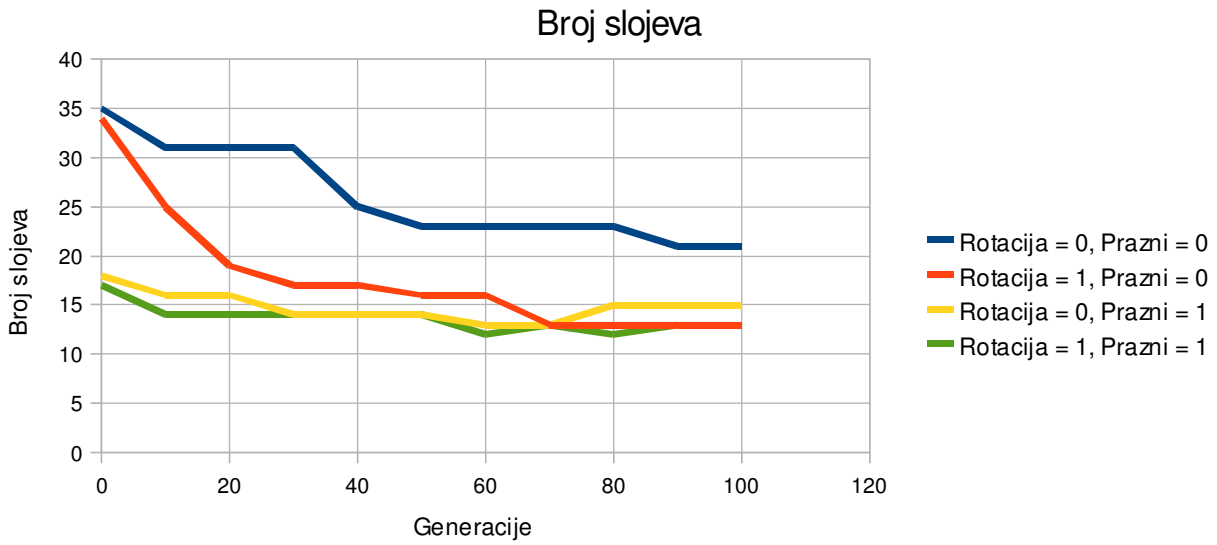
3.3.5 Rezultati simulacije

U simulaciji se početna generacija stvara slučajnim odabirom rasporeda 80 objekata. Širina spremnika je 30. Broj jedinki u svakoj generaciji je 100, vjerojatnost križanja 70%, vjerojatnost mutacije 30%, Broj generacija iznosi 100. Slikom 3.5 su dani grafovi četiri izvođenja koja su dobivena mijenjanjem parametara rotacije i načina stvaranja novih prostora. Kod stvaranja nove generacije, jedan roditelj se bira slučajnim odabirom između 25% najboljih iz prijašnje generacije, dok se drugi bira slučajnim odabirom iz cijele generacije.



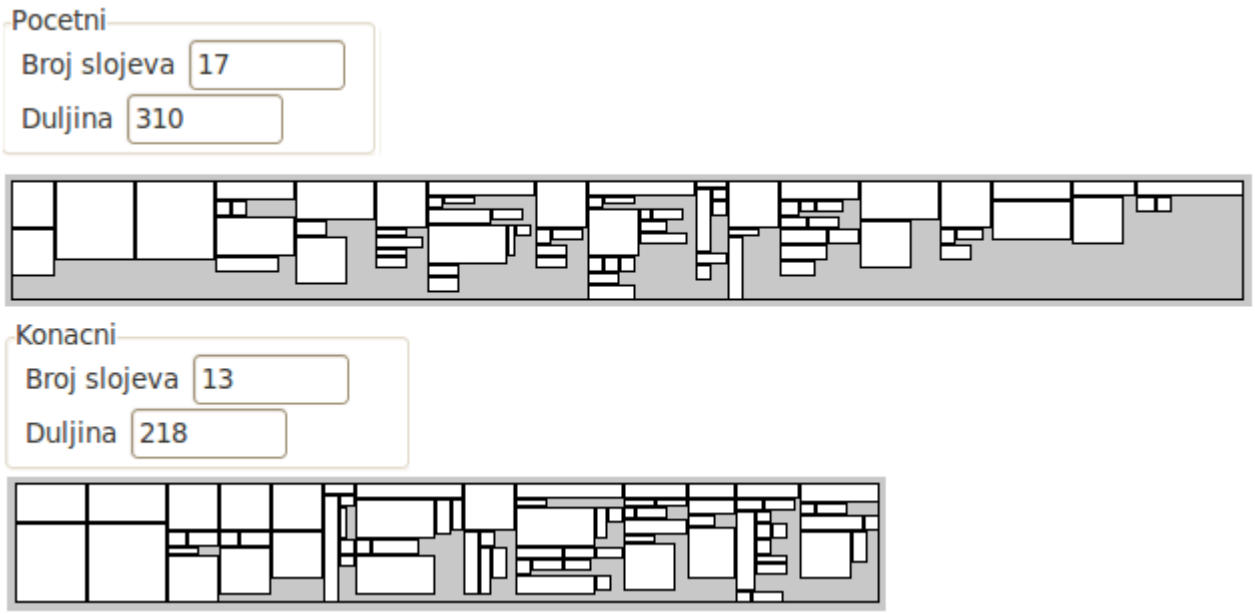
Slika 3.5: Duljine spremnika

Na slici 3.6 prikazan je broj slojeva kroz generacije za sva četiri slučaja.



Slika 3.6: Broj slojeva

Najbolje rješenje dobiva se u slučaju sa vrijednostima parametara *rotacija* = 1 i *prazni* = 1. Izgled rasporeda početnog i konačnog rješenja tog slučaja dan je slikom 3.7

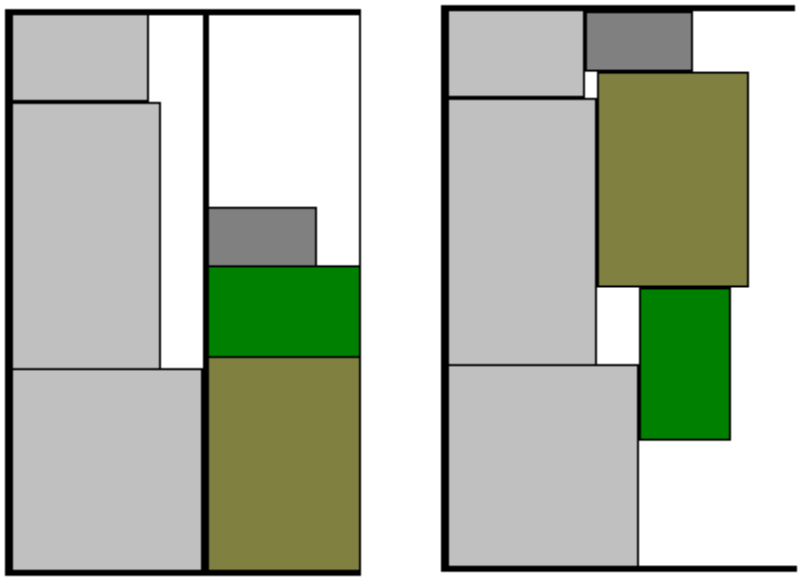


Slika 3.7: Najbolje rješenje

3.3.6 Poboljšanje algoritma

Efikasnost algoritma bi se mogla povećati na dva načina:

- kad bi se kod stvaranja novih praznih mjesta sva prijašnja prazna mjesta spojila, dobila bi se veća prazna mjesta u sloju što bi omogućilo umetanje više objekata
- nakon pronalaska najboljeg rješenje, mogle bi se maknuti granice slojeva te objekti posložiti iz višeg sloja tako da se popune slobodna mjesto u sloju ispod. Primjer spajanja slojeva prikazan je slikom 3.8



Slika 3.8: Spajanje slojeva

4. Zaključak

Genetski algoritmi su moćan alat za rješavanje stvarnih problema koji su vrlo kompliciran za analitičko rješavanje ili analitičko rješenje ne postoji. Raspodjela objekata je jedan od problema za koji ne postoji egzaktno analitičko rješenje. Za malen broj objekata problem pronalaska optimalnog rasporeda je trivijalan no za veći broj objekata problem je vrlo teško riješiti. Najveći problem prikaza rješenja pomoću slojeva je što u svakom sloju ostaje podosta slobodnog mjesta koje se u mnogim slučajevima ne može iskoristiti. Iako se čini da veći boj slojeva doprinosi lošijem rješenju iz prikaza broja slojeva po generacijama uočava se da porast broja slojeva kod nekih rješenja doprinosi boljem rješenju. Velik utjecaj na kvalitetu rješenja imaju parametri rotacije i način stvaranja praznih mjesta. Algoritam raspodjele objekata ima velike primjene u industriji kod pakiranja proizvoda i kod transporta te se se lako može prilagoditi za rješavanje raspodjele u 3 dimenzije.

5. Literatura

Bibliography

(Melanie, 1998): Melanie Mitchell, An introduction to genetic algorithms, 1998

(Kim, 2001): Kim F. Man,K. S. Tang,S. Kwong, Genetic algorithms: concepts and designs, 2001

(Miller, 1995): Brad L. Miller, Genetic Algorithms, Tournament Selection, and the Eects of Noise, 1995

(Andreas, 2003): Andreas Bortfeldt, A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces, 2003

6. Sažetak

Kao uvod u genetske algoritme dan je biološka pozadina ideje. Zatim su opisani osnovi pojmovi vezani uz algoritme, opisane pojedine operacije u postupku izvođenja i početni parametri općenitih genetskih algoritama. U drugom dijelu opisana je konkretna primjena ideje algoritma na rješavanje problema rasporeda objekata u spremnik. Objašnjeni su rezultati izvođenja algoritma i utjecaj parametara na uspješnost izvođenja.