Univerzitet u Beogradu
Matematički fakultet

Seminarski rad

Predrag Mitić

2D Rectangle Packing Problem u kreiranju atlasa

Računarska Inteligencija

Beograd 2024

Sadržaj

Uvod

U gejming industriji jedan od najvećih izazova je učitavanje aseta u što kraćem vremenskom periodu. Da bi se to postiglo, poseže se za raznim trikovima u vidu kompresija aseta ili grupisanjem u veće jedinice. Sve fajlove možemo ukrupnjavati, od tekstualnih, slikovnih, muzičkih pa čak i video fajlova. U ovom tekstu ćemo se fokusirati na grupisanje manjih slika u jednu veliku, takozvani 'atlas'. Atlas je takođe običan slikovni fajl, ali uz njega ide još jedan prateći fajl koji predstavlja neku vrstu mape, odnosno govori nam na kojim koordinatama se nalazi željena slika, kao i koje su joj dimenzije i orijentacija.

Da bi atlas imao što manje dimenzije, potrebno je dobro rasporediti slike u njemu. Pošto su slike pravouganog oblika, naš problem se svodi na problem pakovanja pravouganika, odnsno '2D Rectangle Packing Problem'. Ovaj problem je veoma čest u raznim sferama, kao sto je sečenje materijala tako da otpadak bude što manji, slaganje članaka u novinama, projektovanje čipova, itd.

Rectangle Packing Problem spada u grupu NP teških problema, odnosno ne postoji metoda za rešavnje problema u polinomijalnom vremenu. Zbog toga su rađena mnoga istraživanja i napravljene su razne heuristike i meta-heuristike koje vode do jako dobrih rešenja.

Tragajući za najoptimalnijim rešenjem dvodimenzionalnog pakovanja pravougaonika, u prihvatljivoj vremenskoj i prostornoj složenosti, ispitivane su mnoge metode i poređene međusobno. U tim radovima mogla su se prepoznati dva glavna pristupa rešavanju. Prvi pristup rešava problem primenom heuristika, dok je drugi pristup koristio algoritme zasnovane na meta-heuristikama u kombinaciji sa heurističkim algoritmima, pa smo mogli videti algoritme poput Genetskog Algoritma, Simuliranog kaljenja, Algoritam kolonije mrave itd. koji su se kombinovali sa algoritmima iz prvog pristupa.

U ovom radu cemo koristiti Bottom Left algoritam kao osnovni algoritam,a pokusacemo da njegove rezultate poboljšamo genetskim algoritmom.

1. Problem pakovanja aseta

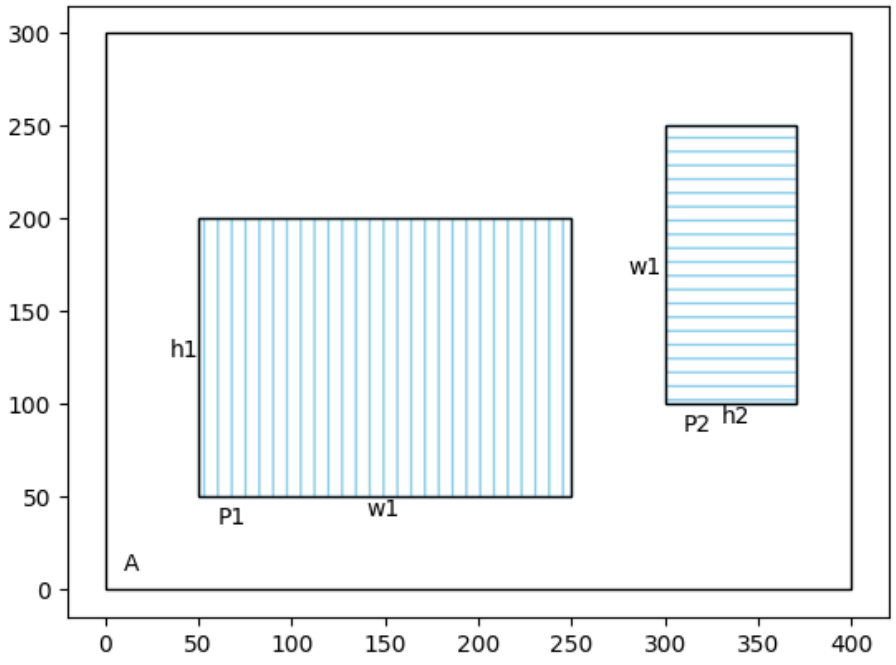
Pakovanje aseta je, kao što je već rečeno, 2D Rectangle Packing Problem, odnosno nadskup tog problema, što podrazumeva da je dozvoljeno menjanje orijentacije pravougaonika, to jest rotiranje slika u ovom kontekstu.

1.1 Definicija problema

Neka je $S = \{P_1, P_2, .., P_n\}$ skup od n različitih slika dimenzija $w_i \times h_i$ za svako $i = 1, 2, .., n$. Treba od njih formirati jednu veliku sliku A maksimalne dimenzije $W \times H$, sa što manjim neiskorišćenim prostorom. Sledeći zahtevi moraju biti ispoštovani da bi pakovanje bilo validno:

1. Nikoje dve slike P_i i P_j se ne preklapaju
2. Dimenzije slike ne smeju biti veće od maksimalnih $W \times H$
3. Slike se mogu rotirati za 90°

Zadatak je naći najmanji mogući pravougaonik, odnosno dimenzije slike A treba minimalizovati, tako da svaka slika P_i bude celovito uključena u konačnu sliku A.



Slika 1.1: Vizuelizacija problema

2. Implementacija algoritama

2.1 Heuristika za slaganje pravougaonika

Jedan od prvih heurističkih algoritama predložila je Brenda S. Baker [1980], pod nazivom Bottom-Left. Glavna ideja ovog algoritma je da se obejkat postavi u gornji desni ugao, a onda se po y-osi spusti najviše sto je moguće, pa zatim i po x-osi se pomeri do krajnje prihvatljive tačke. Kasnije su predstavljena razna poboljšanja i dorade ovog algoritma. Neki primeri su Improved Bottom-Left koji je predložio Dequan Liu [1999]. Ovaj pristup je smanjio procenat neiskorišćenog prostora.

U ovom radu ćemo koristiti taj algoritam (IBL) kao osnova na koju će se nadovezati meta-heuristika i eventualno ga unaprediti u pogledu boljeg iskorišćenja prostora.

IBL algoritam je unaredjeni BL algoritam. Da pojasnim, buttom left algoritam pocinje iz gornjeg desnog ulga, gde pocetno postavlja sliku, onda je spusta najvise sto moze po Y osi, a zatim i po X osi, takodje najvise sto moze. Taj postupak se ponavlja dok ima napretka. Kod IBL korak koji pomera sliku u levo je je modifikovan da sliku ne pomera do kraja, nego samo onoliko koliko je dovoljno da u sledecem koraku može da se pomeri naniže. Ovakav pristup omogućava bolje kretanje slike do finalne pozicije, odnosno sa više potencijalnih pozicija, a složenost ostaje ista.

Složenost i običnog BL i IBL je n^2, jer se za svaku promenu pozicije proveravaju sve ostale slike.

Algoritam 1 Improved Buttom Left algoritam

1: Postavljamo sliku u gornji desni ugao
2: **while** $X_i < X_{i-1}$ ili $Y_i < Y_{i-1}$ **do**
3: Proveri da li je neka slika ispod
4: Pomeri sliku po Y maksimalno
5: Proveri da li je neka slika levo
6: Pomeri sliku po X preko slike iznad koje je ili do sledeće
7: **end while**
8: Vrati poziciju slika

2.2 Genetski algoritam

Genetski algoritam spada u grupu algoriama koji koriste meta-heuristike da bi došli do najoptimalnijeg rešenja. Glavni element u algoritmu je hromozom koji čuva informacije o jedinki, na osnovu kojeg se može oceniti ta jedinka kao i rekonstruisati rešenje iz nje. Jedinke se menjaju ukrštanjem ili mutacijom, a izbor najboljih jedinki se vrši nekom selekcionom metodom. Populacija predstavlja skup jedinki, odnosno hromozoma, koje se iterativno unapređuju i konvergiraju ka optimalnom rešenju.

Jednostavni genetski algoritam (SGA) za reprezentaciju jedinke uzima niz od n bitova, koji su na početku slučajno odabrani, i iz toga pokušava da nadje

optimalno rešenje. Pored osnovne reprezentacije, sledeći bitni faktori su mutacija i ukrštanje. Kod SGA se mutacija dešava na nivou hromozoma, odnosno svako polje u jedinki može mutirati sa malom verovatnoćom P. Ukrštanje je jednopoziciono, odnosno dvema jedinkama se razmene elementi od slučajno odabranog indeksa pa do kraja sekvenci. Selekcija jedinki se vrši ruletski, odnosno svakoj jedinki se da šanasa srazmerna njenom kvalitetu i onda se izaber N jedinki koje će se ukrštati.

U našem slučaju ovaj pristup nije dovaljan, pošto je problem koji rešavamo dosta komplikovan. Imamo dva aspekta koji utiču na ishod primene Bottom-Left algoritma, a to su redosled slaganja i orijentacija slika. Što se tiče orijentacije, pošto nam je jedino važno da li je slika promenila orijentaciju ili nije, tu možemo koristiti reprezentaciju u vidu niza bitova gde ce 0 značiti da slika nije menjala orijentaciju, a 1 da jeste. Drugi aspekt bi bio raspored slaganja slika, a to nije binarni, već kombinatorni problem, odnosno treba naći permutaciju koja ima najbolje karakteristike. Za to ćemo koristiti niz od n celih brojeva, gde će svaki broj predstavljati redni broj postavljanja slike u atlas.

Jedan od pristupa bi bio da jedinke budu sačinjene od uređenih parova (p_i, o_i) za svaki $i \in [1, n]$, gde je p_i pozicija i-tog elementa, a o_i njegova orijentacija. Takav pristup bi bio logičan, ali teži za operacije koje ce biti vršene nad njim. Stoga ćemo razdvojiti p_i i o_i u dva dela niza, koji će sada biti dužine $2n$.

$$G = [p_1, p_2, ..., p_n, o_1, o_2, ..., o_n]$$

Ovakva reprezentacija jedinki nam omogućava da radimo višepoziciona ukrštanja i mutacije, kao i da lako utvrdimo da li je jedinka ispravna ili ne.

Generišemo populaciju od N različitih jedinki. Svaka jedinka se inicijalizuje na nasumičnu vrednost. Pošto se prvi deo genotipa sastoji iz rednih brojeva, oni ne mogu biti generisani nasumično, već se unapred zadati niz $[1, n]$ meša. Za mešanje koristimo funkciju $shuffle()$ iz *random* biblioteke. Drugi deo genotipa je jednostavniji za generisanje. Za svaki objekat generišemo slučajan p iz segmenta $[0, 1]$, ako važi da je $p < 0.5$ objekat ne menja orijentaciju, inače menja.

Sledeći zadatak je utvrditi koliko je neka jedinka kvalitetna. Ako znamo da je potrebno minimizovati dimenzije atlasa, onda je jasno da će kvalitet jedinki biti obrnuto proporcionalan dimenzijama dobijenog atlasa. Dakle, prvo treba primeniti heuristiku pakovanja skupa S nad datom jedinkom i kao rezultat uzeti širinu W i visinu H dobijene slike (atlasa). Odavde imamo da je funkcija kvaliteta jedinke sledećeg oblika

$$fitness(S) = \frac{1}{W * H}$$

Sada imamo na osnovu čega da uporedimo dve jedinke pa možemo razmatrati način na koji ćemo birati koje jedinke će biti iskorišćene za sledeću iteraciju. Treba biti pažljiv sa tehnikom selekcije, jer ako se biraju samo najkvalitetnije jedinke, to pretragu može odvesti ka nekom lokalnom minimumu iz koga se neće izvući pa samim tim i neće doći do optimalnog rešenja. Ova pojava se naziva *selekcioni pritisak*, a on ne treba biti prevelik. S druge strane, ako se potpuno randomizuje proces selekcije onda konstantno gubimo kvalitetne

jedinke. Stoga ćemo koristiti takozvanu *turnirsku selekciju*, koja iz populacije uzima nasumično k jedinki i od njih bira najbolje za sledeću populaciju.

Ostalo je još samo da se izabrane jedinke ukrste i eventualno mutiraju da bismo sastavili novu generaciju. Ukrštanje mora biti višepoziciono jer imamo dva potpuno odvojena dela genotipa koji se ne mogu mešati. Nasumično biramo dva indexa $i \in [1, n]$ i $j \in [n+1, 2n]$. Ako imamo jedinke G_p i G_q, nove jedinke G_{pn} i G_{qn} ce biti formirane na sledeči način: jedinka G_{pn} nasleđuje gene iz segmenata $[1, i]$ i $[n+1, j]$ dok se segment $[i+1, n]$ uzima iz G_q tako da brojevi koji su bili u istom segmentu u G_p dobiju uredjene iz G_q. Segment $[j+1, 2n]$ se prepisuje iz G_q u potpunosti. Jedinka G_{qn} se dobija na isti način samo su drugačije pozicije segmenata.

$$G_p = [1, 2, 3, |4, 5||1, 1, 1, 1, |1]$$
$$G_q = [5, 4, 3, |2, 1||0, 0, 0, 0, |0]$$
$$G_{pn} = [1, 2, 3, |5, 4||1, 1, 1, 1, |0]$$
$$G_{qn} = [3, 2, 1, |4, 5||0, 0, 0, 0, |1]$$

Mutacija se dešava sa malom verovatnoćom P na svakom genu. Ako se desi na prvih n pozicija niza, bira se index $i \in [1, n]$ sa kojim se razmenjuju elmenti, a ako se desi na segmentu $[n+1, 2n]$ vrednost se menja njegovom negacijom. Ovim smo sačuvali ispravnost jedinke, a opet je minimalno izmenili.

Algoritam 2 Genetski algoritam

1: Inicializujemo jedinke
2: **while** Nije dostignut broj generacija **do**
3: Izabor jedinke za sledeću generaciju
4: Ukrstanje svake dve susedne jedinke
5: Pokušaj mutacije na svakoj jedinki
6: Ažuriranje fitness-a na novim jedinkama
7: **end while**
8: Izbor najbolje jedinke

2.3 Iscrpni algoritam

Ako se držimo pretpostavke da Bottom-Left algoritam daje optimalno slaganje algoritma ako mu se da adekvatno uredjen niz pravugaonika, odnosno slika u našem slučaju, možemo zaključiti da će se do optimalnog rešenja sigurno doći ako se prođe kroz sve moguće rasporede, odnosno sve permutacije niza pravougaonika. Za svaku permutaciju biće izračunata površina pokrivajućeg pravougaonika, pa će se raspored koji da najmanju površinu smatrati optimalnim.

Algoritam za generisanje permutacija je urađen prema pseudo kodu iz knjige profesora Stanić [2018].

Napravićemo algoritam koji će ispitati sve moguće permutacije i dati optimalan raspored za Bottom-Left algoritam.

Algoritam 3 Iscrpni algoritam za ređanje slika u atlas

```
1: minDimenzije = infinite
2: trenutnaPermutacija = 1:n
3: poredak = trenutnaPermutacija
4: orijentacije = zeros(n)
5: while trenutnaPermutacija.isSorted(-1) do
6:     novaPermutacija = sledecaP()
7:     while orijentacije.all(1) do
8:         novaOrijentacija = sledecaO()
9:         dimenzije = bottomLeft(novaPermutacija, novaOrijentacija)
10:        if dimenzije < minDimenzije then
11:            minDimenzije = dimenzije
12:            predak = novaPermutacija
13:            orijentacije = novaOrijentacija
14:        end if
15:    end while
16: end while
17: (poredak, orijentacija)
```

3. Testiranje i rezultati

Za testiranje je korišćen laptop koji ima procesor Intel i7-8665U i 32 GB RAM memorije.

Uzećemo set od 6 slika nad kojim možemo da testiramo oba pristupa i jedan malo veći set od nasumično generisanih 20 slika, za koji je iscrpni algoritam neizvršiv.

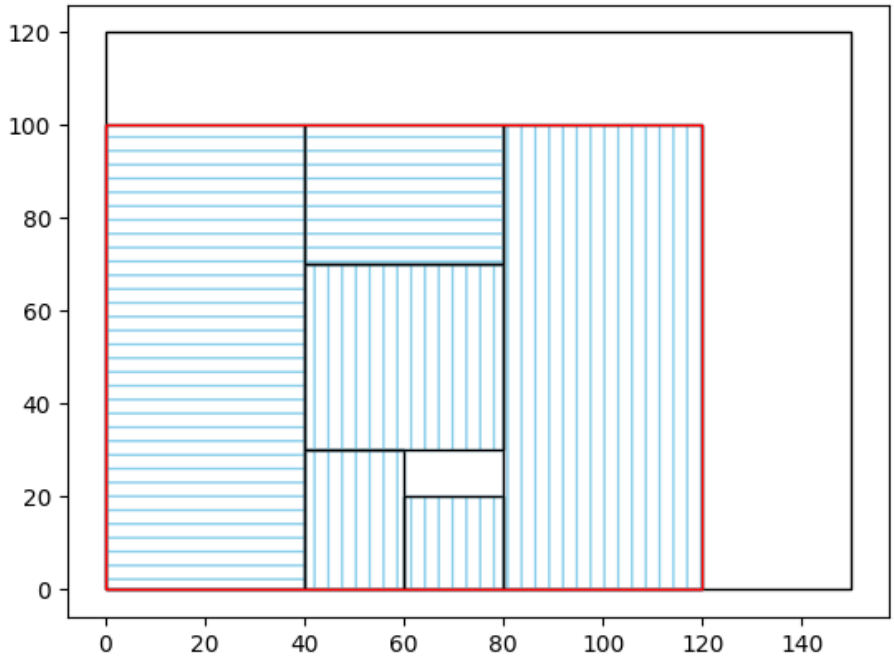
$$Set = [(100, 40), (40, 100), (20, 30), (40, 40), (30, 40), (20, 20)]$$

Slike su predstavljene uredjenim parovima gde prvi broj predstavlja širinu, a drugi visnu slike.

3.1 Testranje Iscrpnog algoritma

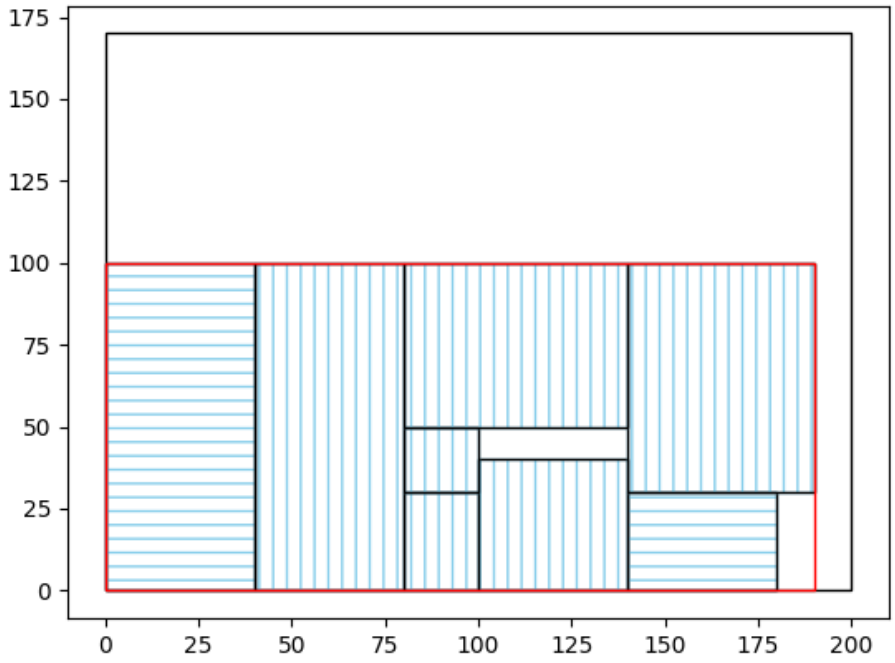
Iscrpni algoritam, kao što je već rečeno, prolazi kroz sve moguće permutacije rasporeda slika i kroz sve kombinacije orijentacije za svaku permutaciju. Kako smo uzeli 6 slika, ima 6! permutacija i za svaku permutaciju još 2^6 orijentacija. Na sve ovo dodajemo složenost heurističkog algoritma za pozicioniranje slika IBL koji ima složenost n^2. Ukupno 1,658,880 operacija da bi se proverile svi mogući rasporedi slika. Ako bismo imali 10 slika taj broj bi bio 371,589,120,000 što je praktično neizvršivo.

Iscrpni algoritam je došao do rešenja za 1.5 sekundi, sa 98.3% iskorišćenosti prostora.



Slika 3.1: Rezultat iscrpnog algoritma za 6 slika

Iscrpni algoritam je došao do rešenja za 604 sekundi, sa 96.3% iskorišćenosti prostora.



Slika 3.2: Rezultat iscrpnog algoritma za 8 slika

3.2 Testranje Genetskog algoritma

Genetski algoritam je testiran sa sledečim parametrima:

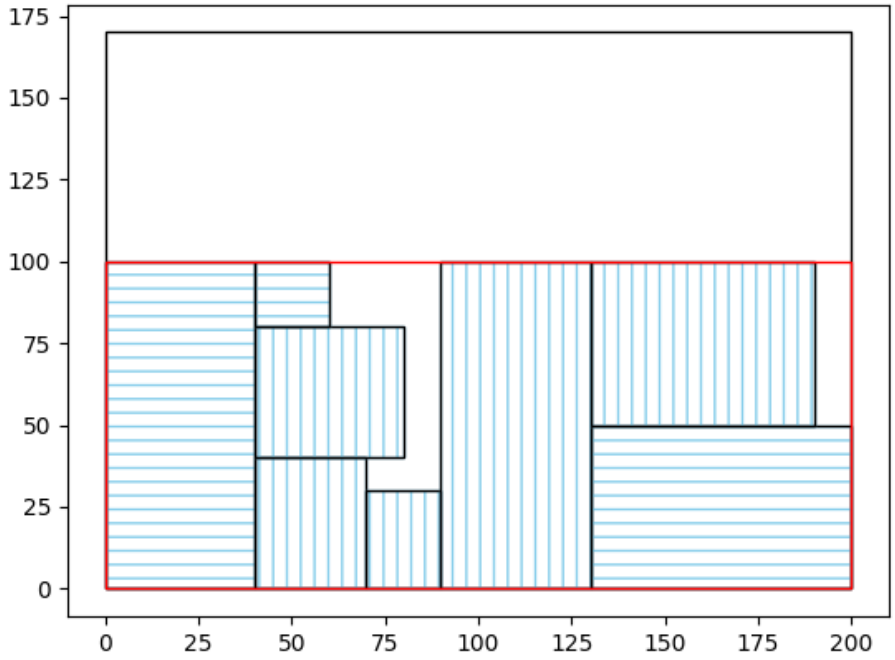
Algoritam 4 Parametri

```
popSize = 75
numGenerations = 150
mutationProb = 0.1
tourSize = 40
elitismSize = 20
```

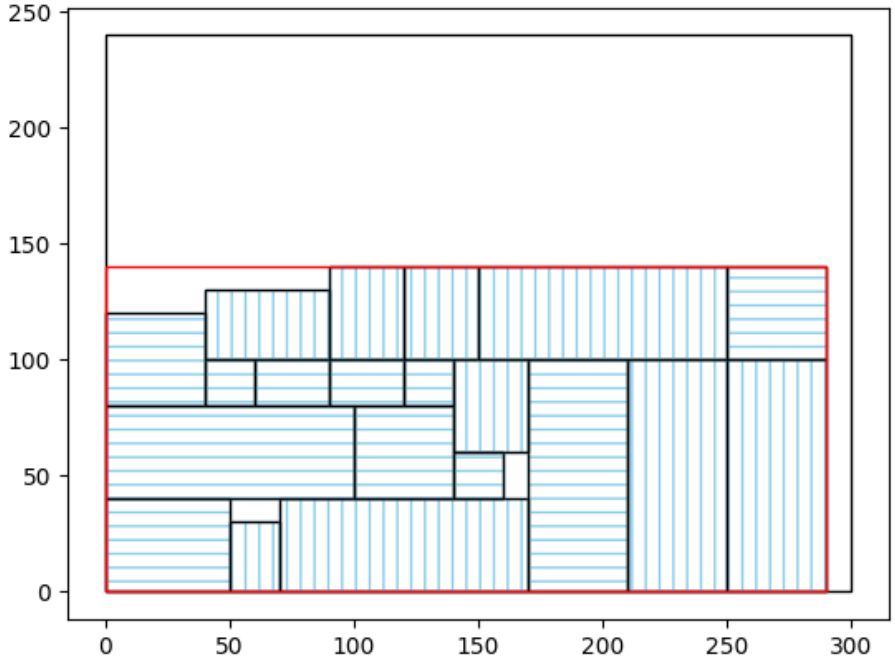
Za početni set podataka genetski algoritam dolazi do istog procenta iskorŕšćenosti za 3 sekunde jer ima zadat fiksan broj generacija.

U ovom slučaju gentski algoritam je dao sledće rešenje, koje ima 91.5% iskorišćenosti prostora. Vreme izvršavanja je bilo ok 4 sekunde.

Za set sa 20 slika, gentski algoritam je dao sledće rešenje, koje ima 95.8% iskorišćenosti prostora. Vreme izvršavanja je bilo ok 5 sekundi.



Slika 3.3: Rezultat genetskog algoritma za 8 slika



Slika 3.4: Rezultat genetsko algoritma za 20 slika

Literatura

Ronald Rivest Brenda S. Baker, Ed Coffman. *Orthogonal packings in two dimensions*. SIAM, 1980. ISBN 846-855.

Hongfei Teng Dequan Liu. *An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles.* EJORES, 1999. ISBN 413–420.

Zoran Stanić. *Diskretne strukture 2*. Matematički fakultete u Beogradu, 2018. ISBN 413–420.