SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 816

RJEŠAVANJE PROBLEMA REZANJA UPORABOM EVOLUCIJSKIH ALGORITAMA

Đorđe Grbić

Zagreb, lipanj 2009.

{Izvornik}

SADRŽAJ

1. Uvod

U modernim industrijskim primjenama često dolazi do potrebe za optimalnom podjelom nekog prostora. Neke od primjena su: drvna industrija, industrija papira, staklarske radionice, tekstilna industrija, gdje se narudžbe manjih komada materijala moraju izrezati iz standardiziranih araka tako da na kraju bude što je moguće manje otpada. U novinskim redakcijama se slažu članci i reklame na standardni format novina tako da nema praznih prostora. Skladištari žele posložiti artikle, koji se najčešće nalaze u kutijama, tako da imaju što manje neiskorištenog prostora. Velik dio tih problema se svodi na optimalno slaganje skupa pravokutnika unutar nekog dvodimenzionalnog prostora.

U ovom radu se opisuju neki od tih problema i načini na koji se oni rješavaju. U opisu će se zadržati na dvodimenzionalnim problemima. Također će biti opisana i istražena uloga genetskih algoritama u rješavanju tih problema, zbog velikog prostora stanja prisutnih kod takvih zadataka. Na posljetku će biti ispitana učinkovitost nekoliko različitih kako pristupa rješavanja problema, tako i načina prikaza i kvaliteta dobivenih rješenja.

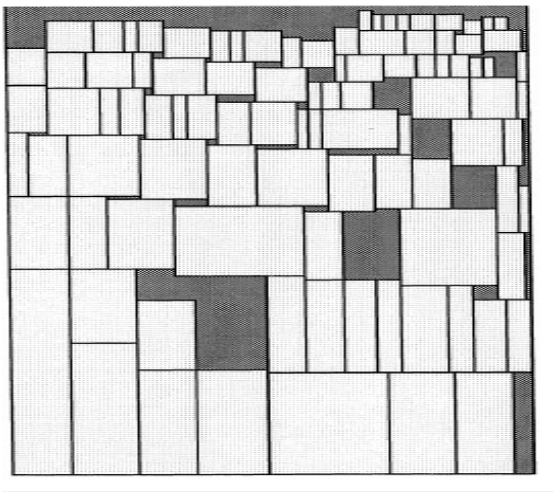
2. Opis problema

2.1. Dvodimenzionalno pakiranje u rolu

Dvodimenzionalno pakiranje u rolu [1] (eng. Two-Dimensional Strip Packing Problem) (2SP prema Lodijevoj notaciji[7]) je definirano na sljedeći način. Za početak imamo rolu materijala koju koristimo za izrezivanje narudžbi. Širina role je unaprijed zadana i iznosi W, dok je duljina role, uvjetno rečeno, neograničena. Širina role je cijelom dužinom jednaka, što znači da je pravokutnog oblika. Zadan je skup pravokutnika širine $w : 0 < w \leqslant W$ i duljine $h : 0 < h$. Cilj je zadani skup pravokutnika smjestiti u rolu tako da ne postoje dva pravokutnika koja se preklapaju i da visina koju tako spakirani pravokutnici zauzimaju bude minimalna. Također se pravokutnici pakiraju tako da im w rub bude paralelan sa W rubom role. Pravokutnici se još mogu okretati za 90°, ali samo oni koji kojima visina h ne prelazi širinu role W, jer u protivnom neće stati na rolu.

2.2. Dvodimenzionalno pakiranje u spremnike

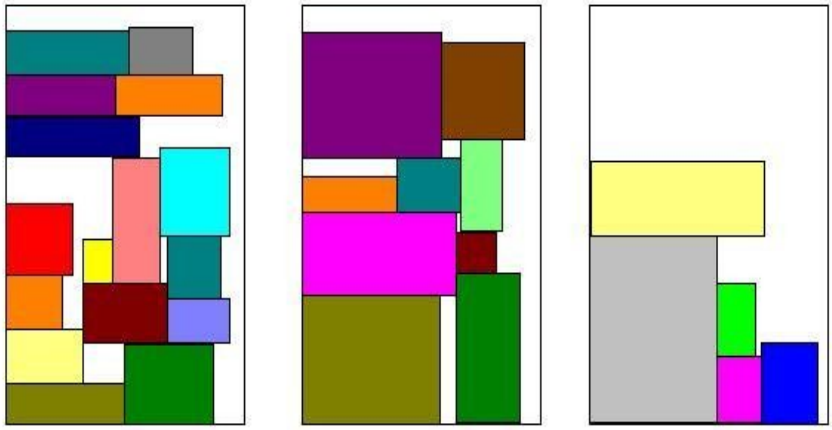
Dvodimenzionalno pakiranje u spremnike [1] (Two-Dimensional Bin Packing Problem) (2BP) je definirano na sljedeći način. Zadan je neograničen broj spremnika. Svi spremnici su jednakih dimenzija. Širina im je W, a visina H. Zadan je i skup manjih pravokutnika koje treba smjestiti u navedene spremnike.



Slika 2.1.

primjer pakiranja u rolu [5]

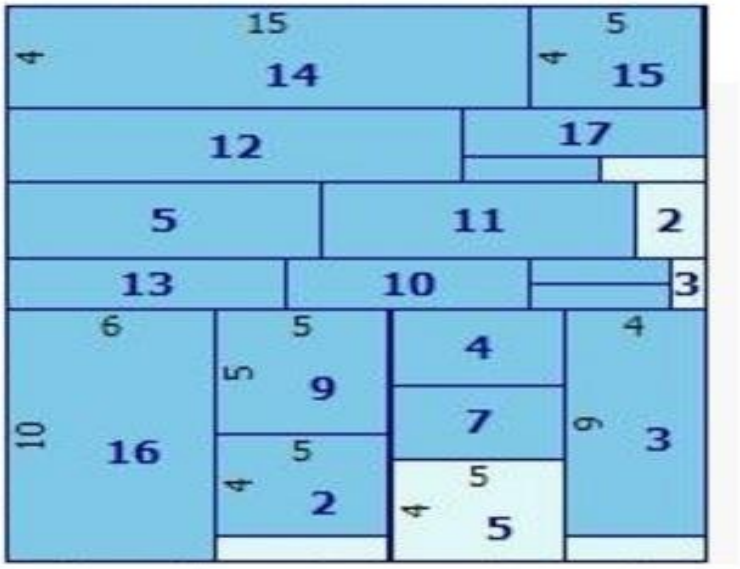
Širine tih pravokutnika su $w_i : 0 < w_i \leqslant W$, a visine su $h_i : 0 < h_i \leqslant H$, gdje i označava redni broj pravokutnika u skupu.Cilj je smjestiti pravokutnike u spremnike tako da ne postoje dva pravokutnika koja se preklapaju i da neiskorišteni prostor između složenih pravokutnika bude što manji. Minimizacija neiskorištenog prostora između složenih pravokutnika za posljedicu ima to da je broj spremnika iskorištenih za pakiranje minimalan (slika 2.2.). U oba problema ćemo pretpostaviti, bez gubitka općenitosti da su ulazni podaci pozitivni cijeli brojevi.



Slika 2.2

pakiranje u spremnike sa slobodnim rezovima

U nastavku su navedeni još neki zahtjevi. Prvi je mogućnost okretanja pravokutnika za 90°. Probleme u kojima se pravokutnici ne smiju okretati zvati ćemo orijentiranima (eng. Oriented). Drugi zahtjev je način na koji izrezujemo manje komade iz većeg. Postoji razlika između giljotinskih rezova (eng. Guillotine cuts) i slobodnih rezova (eng. Free cuts). Giljotinski rezovi su rezovi koji cijepaju arak ortogonalno po cijeloj dimenziji. Takvi rezovi su potrebni jer su strojevi koji režu arke u nekim slučajevima, zbog karakteristika materijala, sposobni rezati samo s kraja na kraj. Jedan od primjera je rezanje stakla gdje se nožem zareže po duljini i onda se prebija preko ruba stola. Primjer jednog takvog slučaja je na slici 2.3.



Slika 2.3.

giljotinski rezovi

Uzimajući sve u obzir imamo osam problema koje obilježavamo na sljedeći način. 2SP|O|G – 2D pakiranje (2D Strip Packing, SP) u rolu sa orijentiranim (O) pravokutnicima i giljotinskim rezovima(G). 2SP|O|F – 2D pak. u rolu sa orijentiranim pravokutnicima i slobodnim rezovima(F). 2SP|R|G – 2D pak. U rolu sa pravokutnicima koji se mogu rotirati(R) i giljotinskim rezovima. 2SP|R|F - 2D pak. U rolu sa pravokutnicima koji se mogu rotirati i slobodnim rezovima. Sljedeće oznake se odnose na 2D pakiranje u spremnike (2D Bin Packing, 2BP): 2BP|O|G, 2BP|O|F, 2BP|R|G, 2BP|R|F.

3. Načini rješavanja problema rezanja

Najjednostavniji problem ovoga tipa je problem jednodimenzionalnog pakiranja u spremnike (1D Bin Packing Problem). Zadatak kod ovog problema je skup zadanih duljina { $x_1, x_2, \ldots, x_i$ } svrstati u podskupove gdje ukupan zbroj duljina svih elemenata podskupa neće prelaziti određenu vrijednost X. Cilj je napraviti što manje takvih podskupova. Težina ovog problema je NP-potpuna. Isto vrijedi i za dvodimenzionalne probleme. Ne čudi, stoga, što su svi algoritmi koji se upotrebljavaju za ove probleme zasnivaju na heuristikama. Najčešće se upotrebljavaju algoritmi zasnovani na pohlepnim heuristikama. Nekoliko jednostavnih algoritama koje ću spomenuti ovdje mogu se svrstati u tri kategorije: jednofazni algoritmi s policama (eng. One-Phase shelf algorithms), dvofazni algoritmi s policama (eng. Two-Phase shelf algorithms), algoritmi koji ne koriste police (eng. Non shelf algorithms). Također će ovdje biti opisani samo algoritmi koji se bave 2BP|O|* problemima. 2BP zato jer su isti algoritmi, uz manje preinake, primjenjivi i za 2SP. Orijentiranost (O) je uvedena kao uvijet jer će se mogućnost okretanja uvesti na razini genetskog algoritma.

3.1. Jednofazni algoritmi s policama

Kod pohlepnih algoritama pravokutnici se smještaju u spremnike redom kojim su zadani. Tim redom se smještaju u police(eng. shelves) tako da širina

svake police bude manja od W. Postoje tri jednofazna algoritma s policama koji će se ovdje obraditi, a to su:

Sljedeće mjesto u koje stane sa skraćivanjem visine (eng. Next-Fit Decreasing Height, NFDH) – Ovaj algoritam uzima trenutni pravokutnik širine w_i i visine h_i , te ga ,ako stane, stavlja na najljevije mjesto na polici. Inicijalno je preostala visina spremnika $H_R = H$. Ako je $h_i < H_R$, onda ako je $\sum_{k=x}^{i-1} w_k + w_i < W$, gdje je x indeks pravokutnika na početku police (inicijalno je 0), stavi pravokutnik na tu policu. Ako je $\sum_{k=x}^{i-1} w_k + w_i > W$, onda skrati H_R za h_{max} , gde je h_{max} visina najvišeg pravokutnika na polici. Stavi trenutni pravokutnik na novu policu. Ako je $h_i > H_R$ onda $H_R = H$ i trenutni pravokutnik stavi u novi spremnik na početak police.

Prvo mjesto u koje stane sa skraćivanjem visine (eng. First-Fit Decreasing Height, FFDH) – Sličan postupak kao i u prethodnom algoritmu. Jedina razlika je ta što svaki put kada se dodjeljuje nova polica, ostatak širine prethodne police i h_{max} stvaraju novi pravokutnik koji označava višak i stavlja se u listu viškova. Za svaki novi pravokutnik koji se namjerava smjestiti u spremnik se prvo pogleda postoji li višak u koji taj pravokutnik stane.

Najbolje mjesto u koje stane sa skraćivanjem visine (eng. Best-Fit Decreasing Height, BFDH) – Jednako kao i u FFDH, samo što se lista viškova sortira tako da

se pravokutnici koji se smještaju u viškove smjeste tako da budu što bliže širini viška.

Rad sva tri algoritma je prikazan na slici 3.1.[6] Broj unutar pravokutnika označava redosljed smještanja u spremnike.

Slika 3.1.

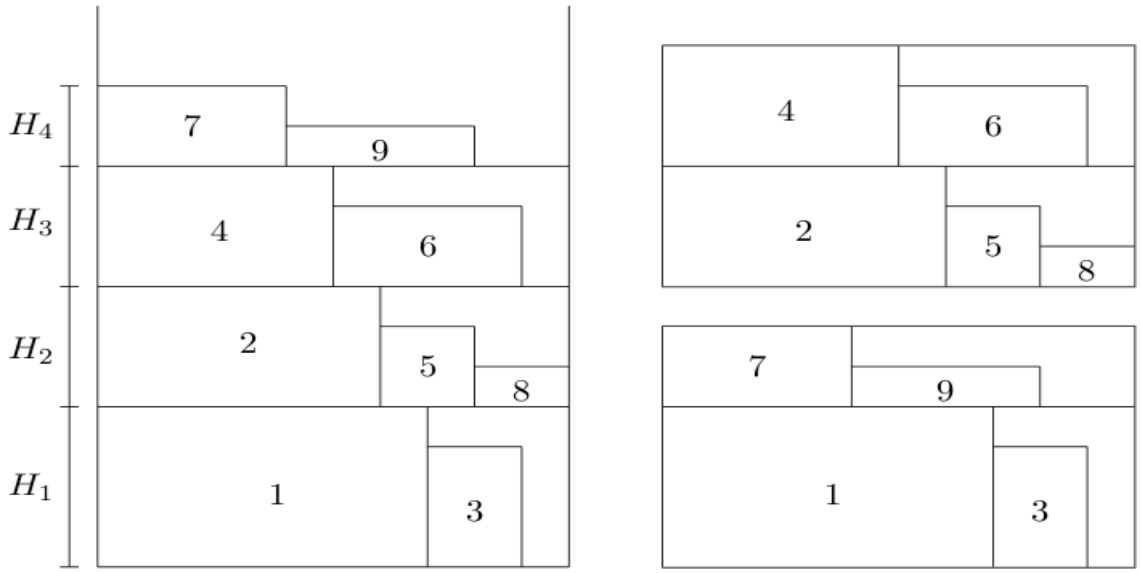
Strategije pakiranja na police

3.2 Dvofazni algoritam s policama

Za dvofazni algoritam s policama mora se modificirati FFDH algoritam tako da se može primjeniti na 2SP problem. Algoritam je u tom slučaju nešto jednostavniji jer nema provjere o dolaženju do vrha spremnika nego se sve police stavljaju na beskonačnu traku.

U prvoj fazi se se odvrti FFDH za 2SP. Zatim se, u drugoj fazi, cijeli problem svede na 1BP problem. Dobijemo skup visina H_i polica koje moramo strpati u jednodimenzionalne spremnike visine H. Opet, slično kao i u FFDH problemu, uzimamo visine redom. Svaku visinu stavljamo u spremnik, ako ne stane uzimamo

novi spremnik i stavljamo ga tamo. Još pazimo da prilikom stavljanja novog pravokutnika pogledamo redom “stare” spremnike i vidimo stanu li, prvo, u njih. Takav algoritam se zove *Hybrid-First Fit* (HFF)[6].



Slika 3.2.

Dvije faze HFF algoritma

3.3 Algoritmi koji ne koriste police

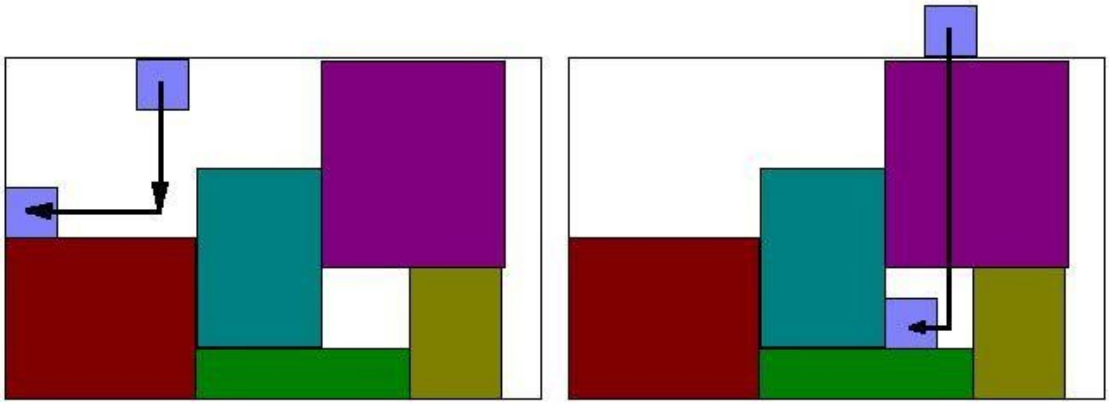
U ovu kategoriju algoritama su svrstana tri algoritma. Dva prilagođena za 2BP|O|F , te jedan prilagođen za 2BP|O|G. Ovi algoritmi su također pohlepni algoritmi jer za svaki novi pravokutnik koji se smiješta određuje trenutnu najbolju poziciju u spremniku.

Algoritmi su sljedeći.

Dno- lijevo algoritam (Bottom-Left algorithm, BL) [2] – Algoritam uzima pravokutnik po pravokutnik, te ga smiješta na najnižu razinu u spremniku. Tada ga

gura na najljevije mjesto na toj razini (slika 3.3. lijevo).

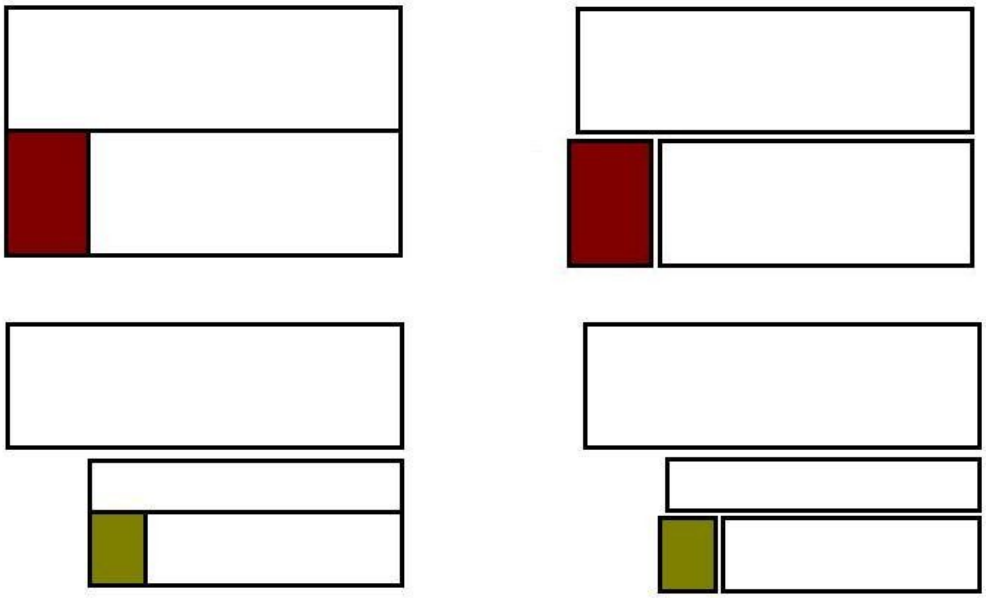
Dno-lijevo ispuni algoritam(Bottom-Left Fill algorithm, BLF) [2] – Algoritam sličan kao i prethodni samo što sada za svaki pravokutnik prvo pogleda može li ga smjestiti u neku od rupa nastalih prilikom slaganja. Ako ne onda primjenjuje BL princip (slika 3.3. desno).



Slika 3.3.

lijevo: BL algoritam desno: BLF algoritam

Algoritam sortiranja giljotinskih ostataka(eng. Sort Guillotine Waste algorithm, SGW) - Zadnji heuristički algoritam uzima redom pravokutnike i pokušava ga smjestiti na prvi ostatak spremnika u listi. Lista je sortirana po površini ostataka uzlazno. Kada smjesti, cijepa ostatak prvo po širini, potom po duljini (slika 3.4.). Ako ne postoji takav ostatak u koji stane pravokutnik, na kraj liste se stavlja novi cijeli spremnik.



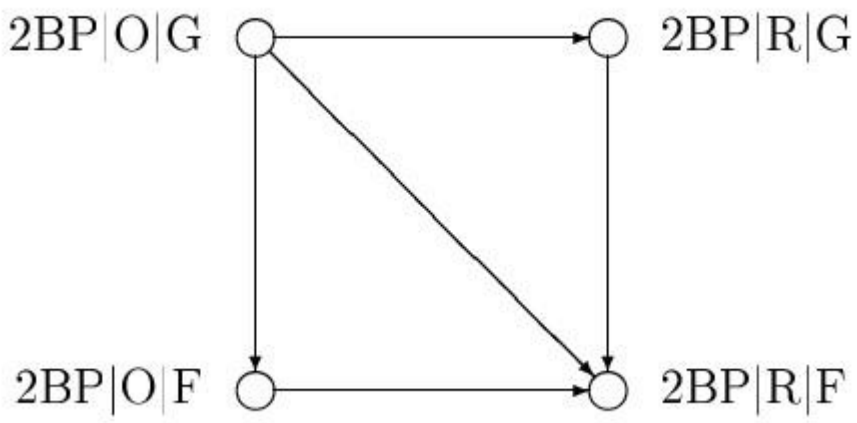
Slika 3.4.

Cijepanje spremnika pomoću SGW algoritma

3.4. Povezivanje algoritama s problemima

Kao što je već rečeno, za svaki od različitih problema postoji algoritam koji ga najbolje opisuje. No neki algoritmi su zbog svojih ograničenja pogodniji za širu lepezu problema. Tako su algoritmi predviđeni za rješavanje 2BP problema pogodni i za rješavanje 2SP problema jer očito spremanje u spremnike je zahtjevnije nego spremanje na rolu. Nadalje, algoritmi primijenjeni na orijentirane probleme (O) su pogodni i za rješavanje problema gdje se pravokutnici mogu rotirati (R). Naravno, tada će takav algoritam davati nužno lošija rješenja jer mu tako automatski sužavamo prostor pretrage. Potonje nas ne brine previše jer će, u daljnjem razmatranju dolaska do rješenja, taj problem nestati. Još su ostali rezovi.

Giljotinski rezovi mogu riješiti i probleme koji ne zahtijevaju giljotinske rezove. Takvi algoritmi također daju lošija rješenja, no zbog svoje jednostavnije implementacije su pogodniji za daljnje razmatranje. Na slici 3.5.[7] je grafom prikazana kompatibilnost između problema i algoritama koji ih mogu riješiti s drugim algoritmima.



Slika 3.5.

Graf odnosa između algoritama i problema

Kružić pored oznake problema označava algoritam koji ga rješava, a strelica upućuje na koji se još isti algoritam može upotrijebiti.

U sljedećem poglavlju ćemo pokušati neke od gore navedenih algoritama učiniti učinkovitijima.

4. Metaheuristički pristup rješavanju problema rezanja

Glavna mana algoritama opisanih u prethodnom poglavlju je što su pohlepni. Pohlepni su jer metode koje primjenjuju prilikom slaganja pravokutnika ne uzimaju u obzir mogućnost uzimanja pravokutnika nekim drugim, možda boljim, redoslijedom. To niti ne čudi jer mogućih redoslijeda ima $n!$, gdje je n broj pravokutnika predviđenih za rezanje, pod uvjetom da nema višestrukih pravokutnika. Još ako se pridoda i činjenica da u problemima gdje se pravokutnici mogu rotirati, veličina prostora naraste na približno $(2 \cdot n)!$. Vidimo da slijepa pretraga prostora nema nekog smisla, a niti se klasične heuristike nisu pokazale pretjerano učinkovitima. Ovdje u priču ulaze metaheurističke metode pretrage prostora. U daljnjoj analizi problema koncentrirati ćemo se na genetski algoritam (eng. Genetic Algorithm, GA).

4.1. Genetski algoritam

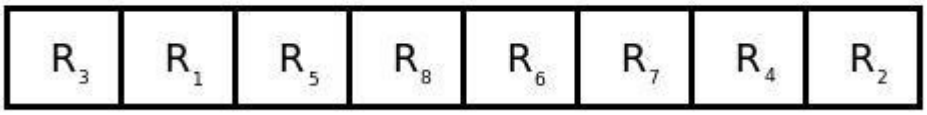
Genetski algoritam je metaheuristička metoda pretraživanja prostora inspirirana biološkom evolucijom. Ovdje ćemo se, za početak, nakratko osvrnuti na opis rada jednostavnih genetskih algoritama. Ideja je da se odredi na koji način će se prikazati općenito rješenje problema. Taj prikaz se zove genotip. Nadalje, stvara se početna populacija nasumično stvorenih rješenja prikazanih pomoću genotipa. Populacija se tada iterativno obnavlja kroz evoluciju. Evolucija podrazumijeva da će

se iz početne populacije izdvojiti genotipovi (jedinke) koje će biti bolje prilagođene okolini. Bolja prilagodba okolini znači da će takve jedinke biti bliže traženom cilju, a to će se odrediti metodama evaluacije genotipa. Takve jedinke se posebnim metodama križaju tako da nastaju "djeca". Geni djece se tada mutiraju kako bi se povećao prostor pretrage prostora stanja. Takva djeca se stavljaju u međupopulaciju, a međupopulacija na kraju svake iteracije evolucije zamjenjuje početnu populaciju. Iteracije se ponavljaju dok se ne ispuni uvjet za prekid algoritma. Proces evolucije je generički postupak i za sve probleme se odvijaju na sličan način. Svakom problemu su specifične metode koje operiraju nad genima jedinki i tip podatka kojim će se predstaviti genotip. Metode su: križanje (eng. Crossover), mutacija (eng. Mutation), evaluacija (eng. Evaluation) . Te metode se zovu genetski operatori (eng. Genetic operators).

4.2. Genotip

Najveći problem kod GA je kako naći najbolji prikaz rješenja, odnosno, genotip. To je slučaj zbog toga što pravilnim odabirom genotipa, već tada, odredimo kakvi će biti ostali genetski operatori. U našem slučaju prikladni genotip je niz pravokutnika koji predstavlja redoslijed slaganja pravokutnika [4]. Pravokutnike ćemo predstaviti pomoću strukture podatka koja se sastoji od dvije cjelobrojne vrijednosti koje predstavljaju visinu i širinu pravokutnika. Još ćemo

definirati metodu koja okreće pravokutnik za 90° tako što ćemo zamijeniti vrijednost visine sa širinom i obrnuto. Cjelobrojnu vrijednost smo uzeli zbog jednostavnijih matematičkih operacija nad njima bez gubitka općenitosti.



Slika 4.1.

Permutacijski vektor pravokutnika R

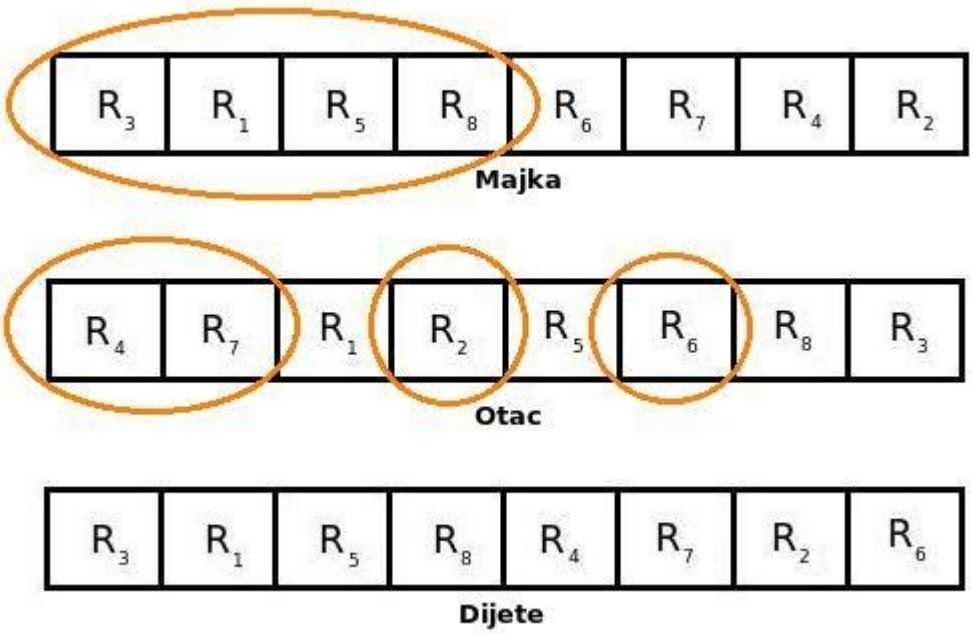
Dakle, genotip će biti permutacijski vektor naručenih pravokutnika R (Slika 4.1.). Svaki sljedeći vektor u populaciji biti će neka od permutacija osnovnog vektora koji se učitava iz datoteke slijedom.

4.3. Križanje

Cilj križanja je da stvori jedinku "dijete" koja će sadržavati obilježja i jedinke "otac" i jedinke "majka". U našem slučaju obilježje jedinke je redoslijed pravokutnika u permutacijskom vektoru. Nameće se pretpostavka da ćemo uzeti podskup vektora iz majke i podskup vektora oca i spojiti u jedinku dijete. No, tada će se najčešće dogoditi da se u djetetu pojave duplikati koji nam uništavaju cijelu koncepciju permutacijskog vektora. Morati ćemo, kod punjenja djeteta iz drugog roditelja, paziti da ne ponavljamo nepotrebno pravokutnike [3]. U tu svrhu ćemo primijeniti tri načina križanja: Pola-pola križanje (eng. Half-Half crossover),

Nasumično-pola križanje (eng. Random-Half crossover) i križanje s nasumičnim podvektorom (Random Sub Array crossover).

Pola-pola križanje uzme podvektor iz majke počevši od indexa 0 do indexa *i*. Index *i* se dobiva sljedećom formulom : $\lfloor i = \frac{i_{max}}{2} \rfloor$, gdje je i_{max} najveći index u vektoru, a znakovi $\lfloor \rfloor$ označavaju funkciju najmanje cijelo. Nakon što smo uzeli gene od majke u sljedećem koraku uzimamo očeve gene redom, preskačući one koji se nepotrebno ponavljaju, i stavlja ih u dijete (Slika 4.2.).



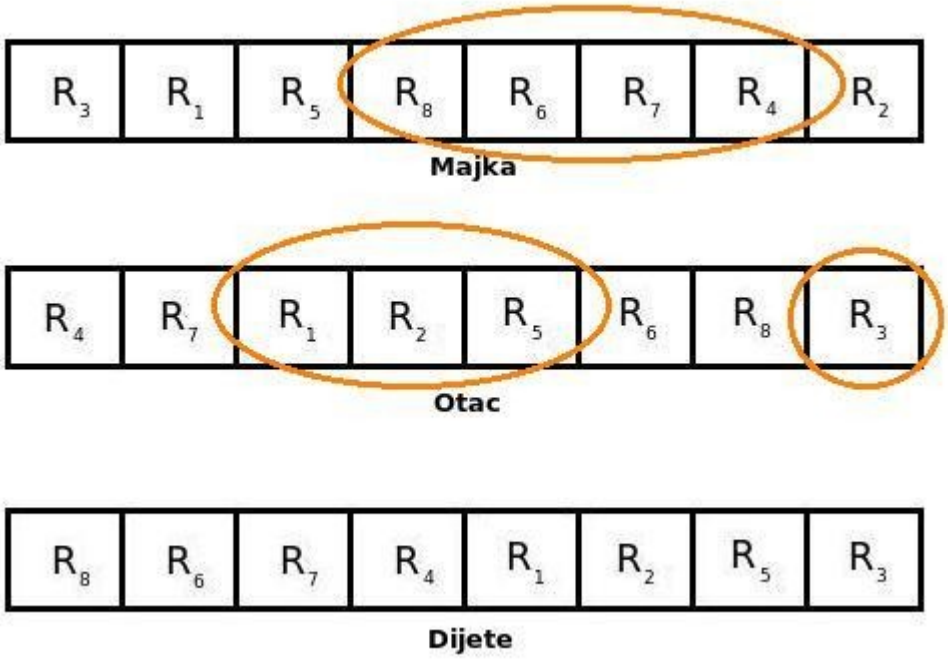
Slika 4.2.

Pola-pola križanje

Nasumično-pola križanje uzme od majke polovicu gena počevši od nasumičnog indexa $i_{begin} : i_{begin} \leqslant \lceil \frac{i_{max}}{2} \rceil$ do indexa $i_{end} : i_{begin} + \lfloor \frac{i_{max}}{2} \rfloor$ gdje znakovi

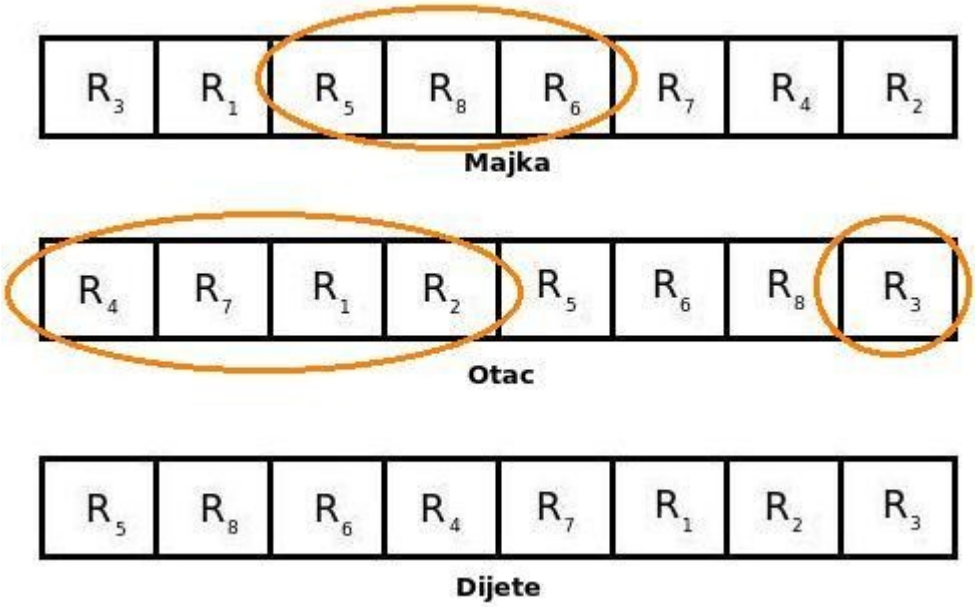
[ ] označavaju funkciju najveće cijelo. Ostatak se uzima od oca tako da se geni nepotrebno ne ponavljaju (Slika 4.3.).

Križanje s nasumičnim podvektorom uzima nasumični podvektor majke počevši od nasumičnog indexa $i_{begin}: 0 \leqslant i_{begin} < i_{max} - 1$ do nasumičnog indexa $i_{end}: i_{begin} < i_{end} \leqslant i_{max}$. Ostatak se napuni kao u prethodnom primjeru (slika 4.4.).



Slika 4.3.

Nasumično-pola križanje



Slika 4.4.

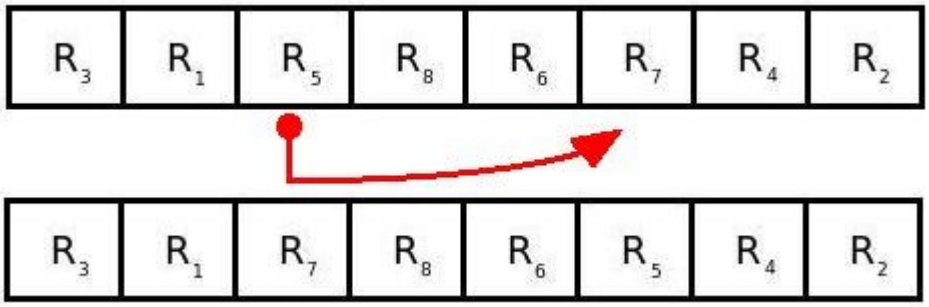
Križanje s nasumičnim podvektorom

4.4. Mutacija

Operator mutacije nakon svakog križanja djetetu nasumično mijenja nekoliko gena. Prije pokretanja algoritma se određuje vjerojatnost mutacije gena. Kada dođe vrijeme za križanje algoritam će za svaki gen u djetetu generirati nasumičan broj od 0 do 1. Ako je taj broj manji od vjerojatnosti mutacije onda se taj gen mutira. Operator mutacije je potreban zato što on proširuje područje pretraživanja stanja. Vjerojatnost mutacije je obično veoma malena jer bi se u protivnom GA sveo na nasumično pretraživanje prostora stanja. Vjerojatnost mutacije gena se, općenito, kreće približno od 0.5% do 2%.

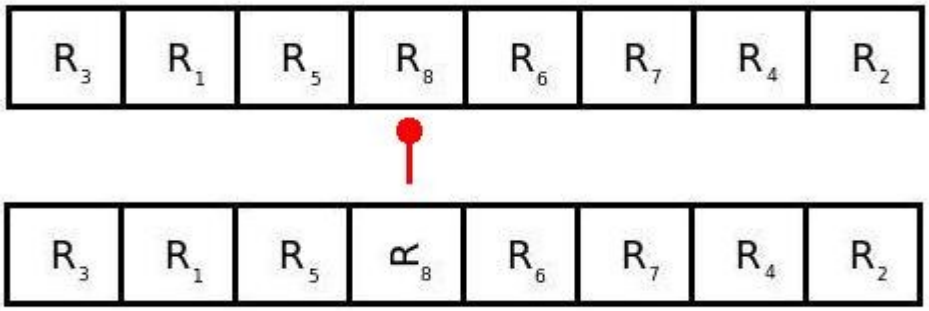
Operator mutacije mora sačuvati smisao genotipa kada ga mutira. Zato će u

našem slučaju operator mutacije raditi dvije operacije. Prvo će za gen koji se odluči mutirati nasumično naći neki drugi gen i s njim zamijeniti mjesto (Slika 4.5.). Ako nam definicija problema dopušta rotiranje pravokutnika za 90°, onda će druga operacija koju će mutacija raditi biti rotiranje odabranog gena (Slika 4.6.).



Slika 4.5.

Mutacija zamjenom mjesta



Slika 4.6.

Mutacija rotiranjem pravokutnika

Kada se obavlja mutacija gena, nasumično će biti izabrano koja će se od dvije mutacije primjeniti.

4.6. Evaluacija

Evaluacija služi da svakoj jedinki odredimo dobrotu (eng. fitness). Dobrota je obilježje svake jedinke. Ona govori koliko je neka jedinka blizu idealnom rješenju.

U našem slučaju broj koji ćemo dodijeliti dobroti će označavati zbroj površina svih višaka koje jedinka stvara. Što je taj broj manji, to je jedinka prilagođenija. Vidimo da je ovdje nespretno upotrijebiti izraz dobrota, te ćemo zbog toga koristiti izraz negativna dobrota (eng. Negative fitness). Višak ćemo računati pomoću heurističkih algoritama koji su opisani u 3. poglavlju. Algoritme koje ćemo uzeti za rezanje su pogodni za rješavanje 2BP|O|G problema. Razlog tome je što su takvi algoritmi prilagođeni za vrlo velik raspon sličnih problema. Orijentiranost ne smeta jer operator mutacije rotira pravokutnike, ako je to dozvoljeno. Od algoritama s policama uzet ćemo samo NFDH algoritam. Razlog tome je što ostali algoritmi troše procesorsko vrijeme na stvari koje bi trebao rješavati GA premještanjem pravokutnika.

Drugi algoritam koji ćemo koristiti je SGW algoritam. Korištenje ovog algoritma je opravdano jer se bitno razlikuje od poličnih algoritama. Algoritam će "izrezati" pravokutnike iz spremnika, izračunati otpad i dodijeliti taj broj jedinki kojoj pripada. Ovim postupkom genotip jedinke pretvaramo u njen fenotip.

4.5. Postupci selekcije

Postupci izbora jedinki iz populacije koje ćemo križati se dijeli u dvije glavne kategorije [9]:

1. Proporcionalne selekcije – Roditelji se biraju nasumično iz populacije ali

vjerojatnost odabira svake jedinke proporcionalna je njezinoj dobroti.

2. Rangirajuće selekcije – Jedinke se prvo rangiraju po dobroti, te se samo najbolje imaju priliku križati.

Od proporcionalnih selekcija izabrati ćemo jednostavnu proporcionalnu selekciju (eng. Roulette Wheel selection). Vjerojatnost odabira jedinke se računa po formuli (4.1.) [9], ako se koristi pozitivna dobrota.

$$p_G(i) = \frac{d_i}{\sum_{n=i}^{N} d_n} \tag{4.1.}$$

U formuli (4.1.) *i* je indeks jedinke za koju se računa vjerojatnost, d_i označava dobrotu jedinke na *i*-tom mjestu, a *N* je broj jedinki u populaciji. Vjerojatnost eliminacije za eliminacijsku proporcionalnu selekciju se računa po formuli (4.2.), ako se koristi pozitivna dobrota.

$$p_E(i) = \frac{d_{max} - d_i}{N \cdot d_{max} - \sum_{n=i}^{N} d_n} \tag{4.2.}$$

U formuli (4.2.) d_{max} je najveća dobrota u trenutnoj populaciji. Kako se u našem slučaju koristi negativna dobrota, za vjerojatnost odabira ćemo koristiti formulu (4.2.). Na intervalu [0,1] dodjeljuje se svakoj jedinki interval duljine vjerojatnosti odabira, tako da se intervali ne preklapaju. Generiraju se dva nasumična broja iz intervala [0,1], te se za roditelje izaberu one jedinke u čije su intervale "upala" ta dva broja. Pritom se pazi da se ne izaberu dvije iste jedinke. Ovaj postupak se

ponavlja dok se ne izgradi sljedeća generacija.

Od rangirajućih selekcija ćemo izabrati 3-turnirsku selekciju. Kod ove selekcije se iz populacije izabiru nasumično tri jedinke, pazeći da su različite. Od odabranih jedinki se dalje izaberu dvije jedinke koje stvore najmanje otpada i križaju.

Koristiti ćemo i elitizam [8]. Elitizam podrazumijeva da ćemo prilikom stvaranja nove generacije obavezno u nju smjestiti i najbolju jedinku iz prošle generacije. Tako čuvamo najbolje rješenje dok ga god ne zamjeni bolje.

Kao uvjet prestanka rada evolucije zadati ćemo broj generacija.

5. Ispitivanje učinkovitosti genetskog algoritma

U ovom poglavlju će se iznijeti eksperimentalni rezultati dobiveni isprobavanjem ranije navedenih operatora i vjerojatnosti mutacija.
Prvo se ispituje učinkovitost operatora.

Operatori korišteni u ispitivanju genetskog algoritma su sljedeći:

Mutacija: Koriste se sve mutacije istovremeno.

Križanja: HHC, RHC, RSAC.

Evaluacije: NFDH i SGW (obilježena kao “Guill.” u tablicama 5.1. i 5.2.) evaluacija.

Selekcije: 3-turnirska (3T) i jednostavna proporcionalna selekcija (Roulette wheel).

Vjerojatnost mutacije gena je, za početak, postavljena na 2%, broj generacija 2000, a veličina populacije je 30 jedinki. Veličina problema je 30 nasumično generiranih pravokutnika, a veličina arka je 1000x1000. Svaka od kombinacija operatora se pokreće 10 puta, te se računa prosjek i disperzija dobrote dobivenih rješenja. Kao kontrolno mjerenje uzeta je nasumična pretraga prostora stanja. Nasumična pretraga prostora se sastoji od generiranja veličina_populacije x broj_generacija (60000) nasumičnih rješenja, te računanja najbolje jedinke pomoću SGW i NFDH evaluacije. Rezultati mjerenja prikazani su u tablicama 5.1. i 5.2. Pokazalo se da HHC križanje, SGW evaluacija i troturnirska selekcija pronalaze najbolja rješenja. Na vrhu tablice su skraćenicama obilježeni operatori upotrijebljeni u ispitivanju. Prvo je križanje, zatim evaluacija, selekcija, vjerojatnost mutacije, broj generacija i veličina populacije.

Generirani pravokutnici su: 555x499, 122x240, 123x469, 612x504, 393x650, 581x39, 473x320, 251x32, 551x481, 531x388, 650x550, 403x323, 398x190, 774x835, 365x799, 375x272, 650x849, 512x125, 318x477, 981x63, 127x562, 455x600, 882x706, 985x786, 539x516, 526x189, 67x281, 865x817, 472x639 i 4x837.

Sljedeće što je ispitivano je utjecaj vjerojatnosti mutacije na kvalitetu rješenja. Iz prethodnog ispitivanja se može zaključiti da HHC križanje, SGW evaluacija i troturnirska selekcija pronalaze najbolja rješenja. Navedena

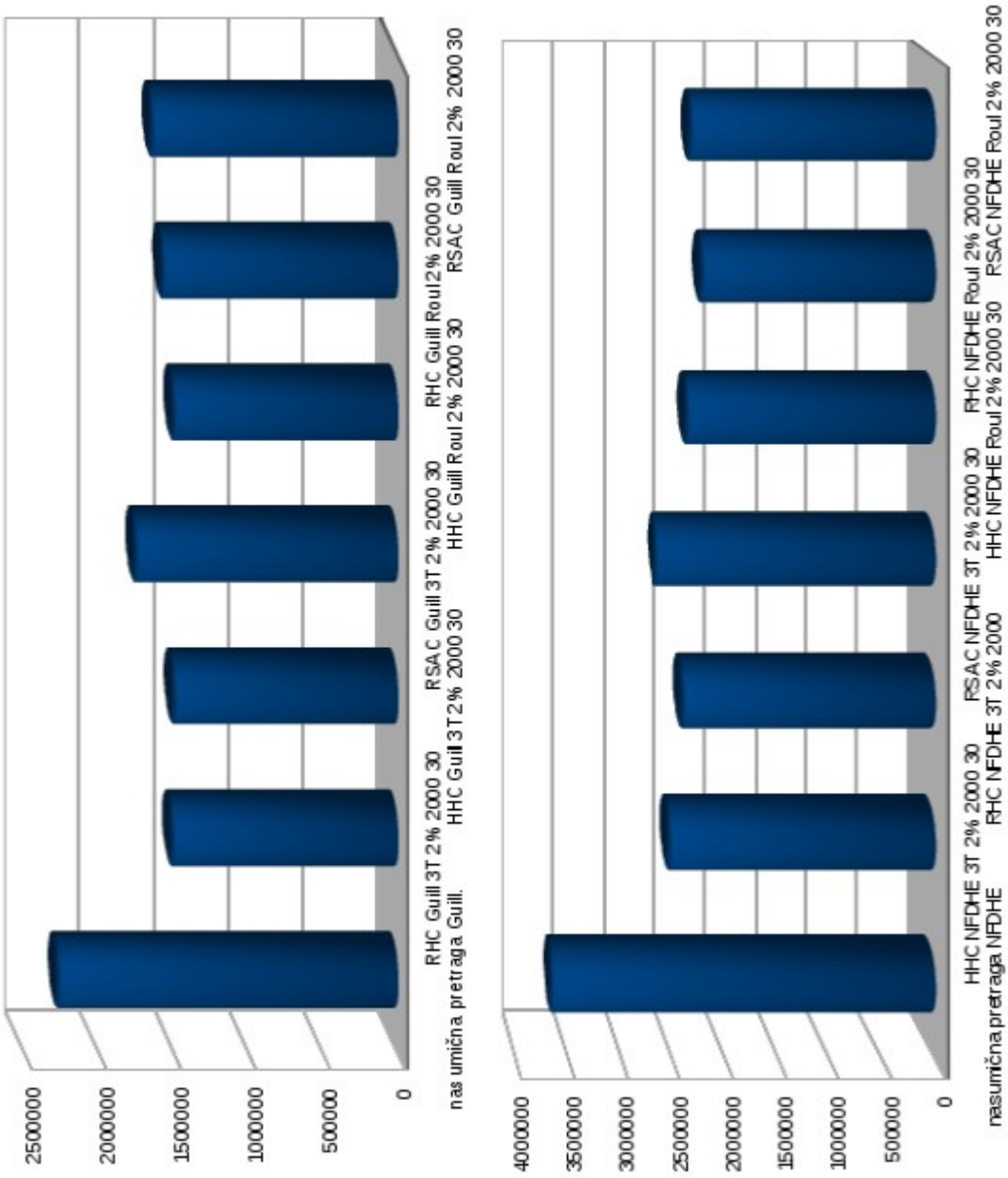
kombinacija operatora je iskorištena u ovome testiranju. Za svaki postotak se algoritam pokreće 10 puta i računa se srednja vrijednost dobrote dobivenih rješenja i disperzija. Rezultati su prikazani u tablici 5.3.

Iz navedenog ispitivanja se vidi da vjerojatnosti mutacije od 3% do 5% daju najbolja i jednako dobra rješenja. Radi preciznijih mjerenja za isti interval je ponovljeno ispitivanje na problemu od 60 pravokutnika, tako da je na postojeći problem od 30 dodano još 30 nasumično generiranih pravokutnika. Rezultati su prikazani u tablici 5.4. Dodani pravokutnici su:814x592, 271x272, 283x687, 520x327, 397x762, 399x940, 846x89, 579x264, 395x563, 927x435, 720x849, 963x102, 516x760, 431x319, 834x20, 586x648, 964x209, 273x247, 896x793, 926x645, 907x326, 585x754, 767x516, 370x162, 79x297, 950x152, 498x913, 254x15, 25x685 i 686x860.

	nasumična pretraga Guill.	RHC Guill 3T 2% 2000 30	HHC Guill 3T 2% 2000 30	RSAC Guill 3T 2% 2000 30	HHC Guill Roul 2% 2000 30	RHC Guill Roul 2% 2000 30	RSAC Guill Roul 2% 2000 30
	2235225	1476225	1476225	2202225	1487225	1487225	1487225
	2292225	1487225	1476225	1487225	1476225	1487225	1487225
	2292225	1487225	1476225	1672225	1476225	1487225	1487225
	2292225	1487225	1476225	1611225	1487225	1487225	1487225
	2292225	1487225	1487225	1543225	1487225	1487225	1487225
	2292225	1487225	1487225	1487225	1487225	1487225	1487225
	2225225	1487225	1487225	2202225	1476225	1487225	1487225
	2225225	1487225	1476225	1487225	1487225	1487225	1476225
	2225225	1487225	1487225	2202225	1487225	1476225	2202225
	2202225	1543225	1476225	1487225	1487225	2202225	2202225
srednja vijednost	2257425	1491725	1480625	1738225	1483925	1557625	1629125
devijacija	34800	10300	5280	278400	4620	128920	229240

	nasumična pretraga NFDHE	HHC NFDHE 3T 2% 2000 30	RHC NFDHE 3T 2% 2000	RSAC NFDHE 3T 2% 2000 30	HHC NFDHE Roul 2% 2000 30	RHC NFDHE Roul 2% 2000 30	RSAC NFDHE Roul 2% 2000 30
	3654225	2611225	2487225	2391225	2309225	2309225	2476225
	3487225	2521225	2202225	2807225	2476225	2202225	2476225
	3719225	2416225	2487225	2611225	2636225	2230225	1780225
	3734225	2491225	2416225	2766225	2292225	2230225	2318225
	3746225	2353225	2392225	2387225	2292225	2309225	2336225
	4027225	2387225	2318225	3230225	2318225	2202225	2336225
	3487225	2543225	2309225	2487225	2292225	2230225	2202225
	3487225	2376225	2202225	2487225	2318225	1814225	2387225
	3672225	2686225	2487225	2487225	2230225	2202225	2341225
	3418225	2674225	2487225	2729225	2230225	2236225	2387225
srednja vijednost	3643325	2506025	2378925	2638425	2339525	2196625	2304125
devijacija	138680	101200	96760	195840	86680	76480	125160

Tablice 5.1. i 5.2.



Grafovi 5.1. i 5.2.

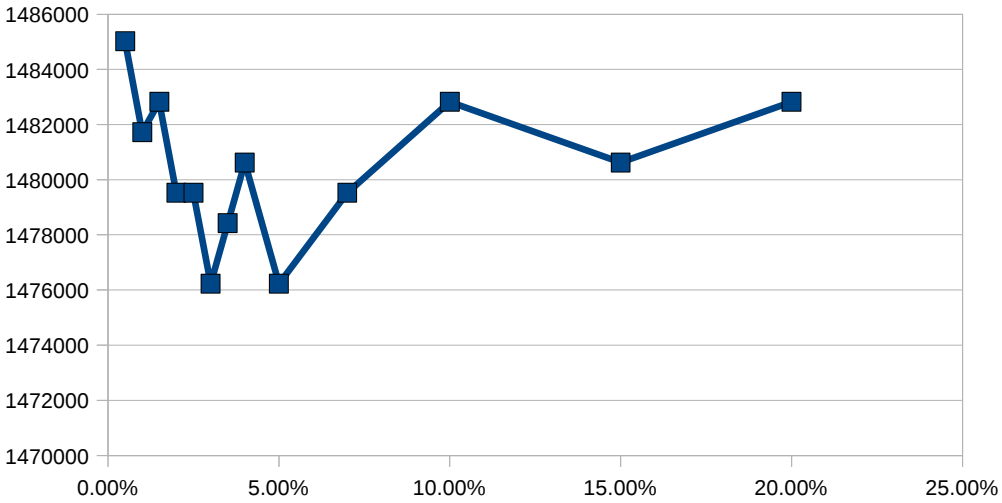
Odnos algoritama i srednje dobrote

najbolih rješenja(Tablice 5.1. i 5.2.)

vjerojatnost mutacije	srednja vrijednost dobrote	disperzija
0.50%	1485025	3520
1.00%	1481725	5500
1.50%	1482825	5280
2.00%	1479525	4620
2.50%	1479525	4620
3.00%	1476225	0
3.50%	1478425	3520
4.00%	1480625	5280
5.00%	1476225	0
7.00%	1479525	4620
10.00%	1482825	5280
15.00%	1480625	5280
20.00%	1482825	5280

Tablica 5.3.

Utjecaj vjerojatnosti mutacije na dobrotu rješenja



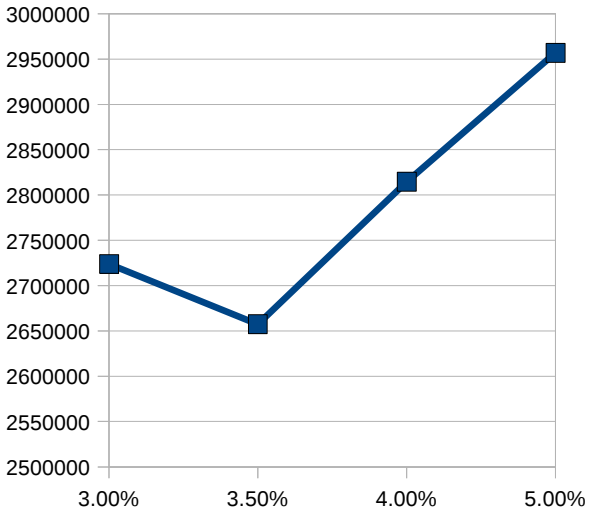
Slika 5.1.

Graf koji prikazuje kvalitetu rješenja ovisnu o vjerojatnoti mutacije (tablica 5.3.)

vjerojatnost mutacije	srednja vrijednost dobrote	disperzija
3.00%	2723826	309840
3.50%	2657426	278720
4.00%	2814526	320300
5.00%	2956926	260720

Tablica 5.4.

Ponovljeno ispitivanje utjecaja vjerojatnosti mutacije na dobrotu rješenja



Slika 5.2.

Graf koji prikazuje kvalitetu rješenja ovisnu o vjerojatnoti mutacije (tablica 5.4.)

6. Zaključak

Rezanje pripada u skup zahtjevnih optimizacijskih problema. Za takve probleme ne postoje egzaktni algoritmi koji ih rješavaju u razumnom vremenu. Kao jedno od mogućih rješenja primjenjuju se genetski algoritmi. Genetski algoritmi ne daju uvijek najbolje rješenje, no često je ono što se dobije tim putem dovoljno dobro. Izbor parametara je presudan čimbenik pri puštanju GA u pogon. Traženje najboljih parametara je veoma zahtjevan posao i presudan za kvalitetu dobivenih rješenja. Još jedan od nedostataka GA su procesorski zahtjevi.

Kod problema rezanja su kao evaluatori korišteni pohlepni heuristički algoritmi NFDH i SGW. Prvi daje lošija rješenja, ali je brži i pogodan je za automatsko rezanje, gdje strojevi prvo izrežu materijal vodoravno, zatim okomito. SGW algoritam daje najbolje rješenje ali je mnogo sporiji jer nakon svakog rezanja mora sortirati ostatke. Half-Half križanje je najbolje ako se upotrebljava SGW evaluacijski operator, a Random-Half križanje je najbolje sa NFDH operatorom. RSA križanje se pokazalo kao neučinkovito. Za vjerojatnost mutacije je najbolje ako se kreće oko 3%.

7. Literatura

[1] Lodi A., Martello S., Monaci M. Two dimensional packing problems: A survey. European Journal of Operational Research, 141 (2002), 241–252.

[2] Hopper E., Turton B.C.H. A Genetic Algorithm for a 2D Industrial Packing Problem. Computers and Industrial Engineering, vol. 37/1-2 (1999), 375-378.

[3] Falkenauer E. A Hybrid Grouping Genetic Algorithm for Bin Packing .

[4] Hinterding R., Juliff K. A genetic algorithm for stock cutting: an exploration of mapping schemes. Technical Report 24 COMP3 (1993) (Draft 1) , 5-6.

[5] Hopper E., Turton B.C.H. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operational Research 128 (2001) 34-57.

[6] Lodi A. Algorithms for Two-Dimensional Bin Packing and Assignment Problems. Dottorato di Ricerca in Ingegneria dei Sistemi. Universita di Bologna . 1999. 3-8.

[7] Lodi A., Martello S., Vigo D., Heuristic and Metaheuristic Approaches for a Class of Two-Dimesional Bin Packing Problems. INFORMS Journal on Computing. Volume 11, Issue 4 (April 1999) 345 – 357.

[8] Golub M. Genetski algoritam: prvi dio. Skripta (2004). 8-20.

[9] Golub M. Genetski algoritam: drugi dio. Skripta (2004). 22-33.

8. Sažetak

Problem rezanja je NP-potpun problem koji se često javlja u industriji. Dobro rješavanje tog problema ima za posljedicu golemu uštedu materijala. Osmišljeni su mnogi algoritmi koji rješavaju ovaj problem i većina ih se zasniva na pohlepnim heuristikama. Bitan napredak na području NP-potpunih problema dogodio se primjenom metaheurističkih metoda. Posebice se ističe genetski algoritam. Prilikom rješavanja ovog problema korišten je hibridni algoritam koji kombinira genetski i pohlepne heurističke algoritme kao evaluacijske operatore. U ovom radu je ispitana učinkovitost nekoliko genetskih i evaluacijskih operatora. Genetski algoritam se pokazao kao učinkovita metoda u rješavanju ovog problema.

Ključne riječi: Rezanje, pakiranje u spremnike, pakiranje u rolu, pohlepna heuristika, genetski algoritam, evaluacija, križanje, mutacija.

9. Summary

Cutting-stock problem is a NP-complete problem with vast implementation in industry. Good solutions of this problem implies great savings of materials. Many of invented algorithms, for solving this problem, has been invented based on greedy heuristics. Significant advancement on field of NP-hard problems happened when implementation of metaheuristic methods began, especially genetic algorithm. Implementation of solution for this problem uses hybrid method algorithm which combines genetic and greedy heuristic algorithms as evaluation operators. This paper contains testing results for solutions given by several genetic and evaluation operators.

Keywords: Cutting-stock, bin packing, strip packing, greedy heuristics, genetic algorithm, evaluation, crossover, mutation.