UDK: 519.863(047)=861
COSATI: 12-01

Heuristička optimizacija razmeštanja pravougaonika na polubeskonačnoj traci zadate širine

Dr Zoran Drašković, dipl.meh.[1)]

U radu se iznosi jedan heuristički postupak pseudooptimalnog razmeštanja proizvoljnog broja pravougaonika u polubeskonačnoj traci zadate širine. Osnovna odlika programa dobijenog implementacijom takvog postupka jeste da daje zadovoljavajuće rezultate i to veoma brzo.

Ključne reči: **optimizacija, heuristička optimizacija, alokacija, „nesting", ploter, polubeskonačna traka, pravougaonik.**

Korišćene oznake i simboli

NMAX, *N* *NMAX*	– maksimalan broj crteža,
H	– zadata širina polubeskonačne trake,
FUGA	– zadato najmanje rastojanje između pravougaonika,
PORA,SPIC	– "iskustveni" parametri koji utiču na poravnavanje, odnosno ravnomerno napredovanje "fronta",
A(NMAX), B(NMAX) *X(NMAX), Y(NMAX)*	– dimenzije crteža iz *queue*-liste i koordinate njegovog središta,
XX(NMAX), XXT(NMAX) *YY(2,NMAX), YYT(2,NMAX)*	– nizovi koji definišu segmente u "frontu", tj. granicu prethodnog i tekućeg "reda",
NSEG, NSEGT	– broj segmenata u prvom, odnosno tekućem "redu",
NRECT	– stvarni broj crteža koji se crta u tekućem "redu",
WIDTH_MAX	– najveća dimenzija crteža koji se crta u tekućem "redu",
AREA_RECT	– ukupna površina svih pravougaonika koji će biti nacrtani,
PROCENAT	– procenat neiskorišćenosti pravougaone crtaće površine

U v o d

OVDE opisani postupak nastao je (pre više od deset godina) u cilju smanjenja nepotrebne potrošnje skupe uvozne hartije prilikom crtanja na ploteru (plotter).

Zadatak, koji je trebalo da se reši, može da se formuliše na sledeći način: potrebno je da se izvrši takvo smeštanje skupa crteža, tj pravougaonika $[a_i, b_i]$ (i=1,2,...,N) , na crtaću površinu plotera, tj. u polubeskonačnu traku zadate širine, da ono bude optimalno u smislu smanjivanja neiskorišćenosti te površine (pri tome ivice pravougaonika treba da ostanu paralelne ivicama trake; više je nego jasno da se pravougaonici ne smeju međusobno "prekrivati").

Međutim, budući da u vreme kada je taj zadatak postavljen nismo imali na raspolaganju programske pakete za optimizaciju kojima bi se takvi problemi rešavali, izvršena je samo *pseudooptimizacija* raspoređivanja crteža na ploteru. Pseudooptimizacija se sastojala u razvoju jednog *heurističkog* postupka*) optimizacije razmeštaja pravougaonika na polubeskonačnoj traci zadate širine.

U radu će biti iznete osnovne odlike takvog algoritma. Takođe će biti navedeni i primeri koji ilustruju mogućnosti implementiranog heurističkog postupka.

Heuristički postupak optimizacije razmeštaja pravougaonika na polubeskonačnoj traci zadate širine

Algoritam, koji će biti upotrebljen za heurističku optimizaciju rasporeda pravougaonika na polubeskonačnoj traci zadate širine, se u osnovi sastoji uređanju pravougaonika u pravcu širine trake, ali tako da se u raspoloživi deo širine trake postavlja onaj od preostalih pravougaonika s najvećom stranicom koja u taj deo može da stane**). Postupak sačinjavaju sledeći koraci:

- biranje prvog pravougaonika,
- dopunjavanje prvog "reda" ***) (pri čemu "redom" smatramo niz uzastopno poređanih pravougaonika od jedne do druge ivice trake),
- popunjavanje ostalih "redova" (dopunjavanjem prvog, a i svakog sledećeg, "reda" dobija se izlomljeni "front" u/uz koji dalje treba "udevati" preostale pravougaonike;

[1)] Vojnotehnički institut VJ, 11000 Beograd, Katanićeva 15

*)Tj. postupka koji se mnogo više odlikuje domišljanjem nego li dometom upotrebljenog matematičkog aparata.

**)To bi se najkraće moglo opisati kao postupak za „udevanje" („ugnježđivanje", „pripijanje") pravougaonika.

***)Ni leva ni desna ivica „reda" (izuzev leve ivice prvog „reda") u opštem slučaju nije glatka, već izlomljena linija sačinjena od „segmenata"− duži upravnih ili paralelnih ivicama trake, koje predstavljaju stranice ili delove stranica poređanih pravougaonika. Desna ivica „reda" predstavlja „front"−granicu do koje je došlo toređanje.

razvijene su i određene korektivne procedure koje sprečavaju da se javi "front" od preteranog broja "segmenata"),
- eventualna korekcija poslednjeg "reda".

Prvo će biti navedene neke opšte napomene vezane za odgovarajući programski kôd. Najpre se vrši učitavanje podataka o crtežima koje treba optimalno rasporediti na polubeskonačnoj traci zadate širine: to su zadati broj crteža N, dimenzije tih crteža iz *queue*-liste $A(i)$ i $B(i)$, potom zadata širina trake H, zadato najmanje rastojanje između pravougaonika *FUGA* i na kraju "iskustveni" parametri *PORA* i *SPIC* koji utiču na poravnavanje, odnosno ravnomerno napredovanje "fronta". Posle završenog "sortiranja" pravougaonika, kontroliše se njihovo međusobno nepresecanje i neizlaženje izvan trake (ovo proveravanje je svoju glavnu ulogu odigralo u fazi testiranja samog programa); najzad, kao neka vrsta kontrolnog parametra, sračunava se i veličina PROCENAT, koja predstavlja procenat neiskorišćenosti crtaće površine (o tome će biti više reči kasnije). Treba napomenuti i da postoji mogućnost da se na alfanumeričkom terminalu prikaže šema razmeštaja crteža u polubeskonačnoj traci zadate širine; to će biti iskorišćeno kasnije, pri komentarisanju nekih primera takvog pseudooptimalnog rešenja.

Biranje prvog pravougaonika

Prvi korak algoritma sastoji se u tome da se od svih pravougaonika izabere pravougaonik sa *najvećom* stranicom koja može da stane u traku zadate širine (uz eventualnu rotaciju pravougaonika za $\pi/2$). Kad se odredi taj pravougaonik, prelazi se na dopunjavanje prvog "reda" dok se ne iscrpi širina trake (v. sledeći odeljak).

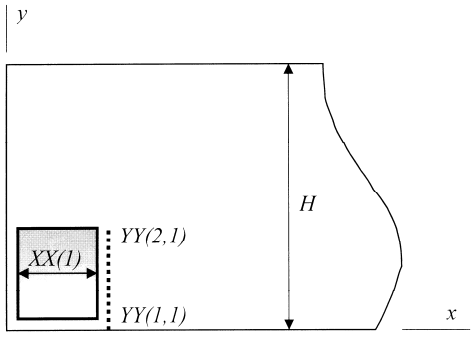
Što se tiče same programske realizacije ovog algoritamskog koraka, uvedeni su nizovi *XX(NMAX)* i *YY(2,NMAX)* u koje se − nakon što je u delu programa

```
WIDTH_MAX=0
...
do i=1,N
        if(B(i).le.H) then
                if(B(i).gt.WIDTH_MAX) then
                        WIDTH_MAX=B(i)
                        NRECT=i
                endif
        endif
        if(A(i).le.H) then
                if(A(i).gt.WIDTH_MAX) then
                        WIDTH_MAX=A(i)
                        NRECT=i
                endif
        endif
  enddo
```

određen (uz eventualnu rotaciju) pravougaonik $[A(i),B(i)]$ sa najvećom stranom *WIDTH_MAX* koja može da stane u traku zadate širine H − smeštaju podaci koji određuju "front", zapravo njegov "segment" koji čini stranica pravougaonika upravna na ivicu trake, a udaljenija od početka polutrake (v. sl.1)*)

```
YY(1,1)=0.
YY(2,1)=WIDTH_MAX
XX(1)=A(NRECT)+B(NRECT)-WIDTH_MAX
```

Pri tom se vrši i ispitivanje da li je neki od zadatih crteža (ili čak svi!) preveliki za zadatu traku i eliminiše se iz liste zadatih crteža.



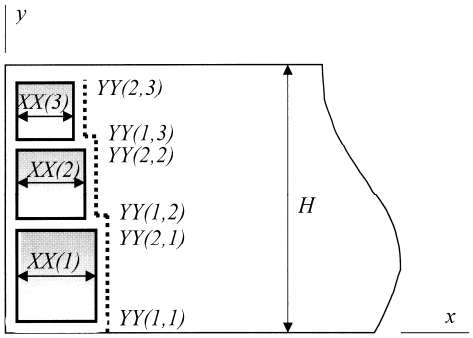
Slika 1.

Za tako izabrani prvi pravougaonik beleži se njegov stvarni, "ulazni" broj *NRECT* (tj. broj crteža u *queue*-listi), kao i njegov status — da li je rotiran ili nije, s tim što se prethodno određuju i koordinate središta odgovarajućeg crteža koji će biti prvi nacrtan u zadatoj polutraci

```
Y(NRECT)=0.5*YY(2,1)
X(NRECT)=0.5*XX(1)
```

Dopunjavanje prvog "reda"

U preostali prostor između prvog pravougaonika i "gornje" ivice polutrake (v. sl.2) treba smestiti "najveći" mogući preostali pravougaonik. Međutim, i kad se to učini, može da preostane još prostora, pa se taj postupak ponavlja sve dok ne preostane "džep" u koji se ne može "udenuti" ni jedan od preostalih pravougaonika; tada treba preći na drugi "red" (o tome će biti reči u sledećem odeljku).



Slika 2.

Što se same programske realizacije tiče, opet se koriste isti nizovi *XX(NMAX)* i *YY(2,NMAX)* da se svaki put, posle nalaženja pravougaonika sa najvećom stranom, koja može da stane u preostali prostor u Y-pravcu, "pamte" karakteristične tačke "fronta", odnosno svakog "segmenta" koji definiše taj "front"

```
YY(1,NSEG)=YY(2,NSEG-1)
YY(2,NSEG)=YY(1,NSEG)+WIDTH_MAX
XX(NSEG)=A(NRECT)+B(NRECT)-IDTH_MAX
```

gde je *NSEG* broj tekućeg segmenta u prvom "redu" (a po završenom dopunjavanju to će biti ukupan broj "segmenata" u prvom "redu"). Naravno, i sada se vodi računa o tome da li je pravougaonik bio rotiran ili ne, a određuju se i koordinate centra pravougaonika

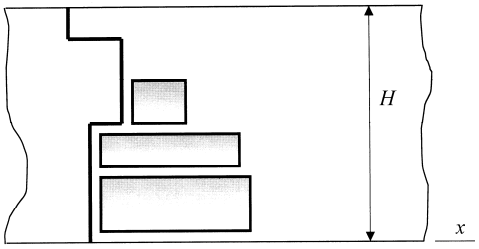
*) Na ovoj, kao i na ostalim slikama u radu, nizom simbola „▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪" biće označavan „front"−granica do koje se došlo u rešenju pravougaonika.

Y(NRECT)=
=YY(2,NSEG-1)+0.5(YY(2,NSEG)-YY(2,NSEG-1))*
*X(NRECT)=0.5*XX(NSEG)*

gde je *NRECT* stvarni ("ulazni") broj crteža koji će biti nacrtan na *NSEG*-tom mestu u prvom "redu". Usput se uspostavlja i korespondencija između broja crteža u *queue*-listi (*NRECT*) i rednog broja (*NSEG*) posle "optimizacije".

Popunjavanje ostalih "redova"

Ovaj deo algoritma sastoji se od onolikog broja ciklusa koliko još ima, tj. koliko će još da bude "redova" i završava se tek kada su "iscrpljeni" svi zadati pravougaonici, tj. crteži. Jedan ciklus sastoji se u tome da se uz "segmente" prethodnog "fronta" (to je desna granica prethodnog "reda") ređaju preostali pravougaonici, ali tako da se uz "tekući" "segment" fronta "priljubljuje" pravougaonik maksimalne dimenzije koja tu može da stane; razume se, može se desiti da je "segment" veći od stranice koja mu se "priljubljuje", tj. da ostaje neki "džep" koji se može takođe popuniti nekim manjim pravougaonikom*) (v. slika 3) ili čak s nekoliko pravougaonika (ukoliko se desi da postoji veći broj i relativno manjih crteža); zatim se prelazi na sledeći "segment", sve dok se ne dođe do gornje ivice trake, tj. dok se ceo "red" ne popuni.

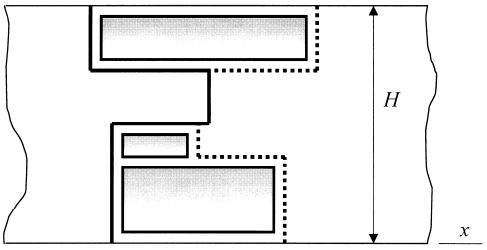


Slika 3.

Takvim postupkom se novi "front", tj. desna granica novog "reda" može učiniti još usitnjenijom, tj. sačinjenom od većeg broja "segmenata" manje dužine. To zahteva da se vrši i određeno poravnavanje "fronta", tj. "zatvaranje" onih "džepova" koji ne mogu da se popune nekim od preostalih pravougaonika. Stoga je razvijena korektivna procedura koja sprečava da se javi "front" od preteranog broja "segmenata"; takvo ažuriranje (poravnavanje) "fronta" zapravo je ključni deo faze algoritma za optimiziranje razmeštaja crteža o kojoj se govori u ovom odeljku; ono je, uz još neke korekcije, osnovni preduslov za kvalitet predložene heurističke optimizacije, tj. za dobijanje "optimalnog" rešenja i nešto kasnije će biti opisano na mogućim slučajevima pri "slaganju" pravougaonika.

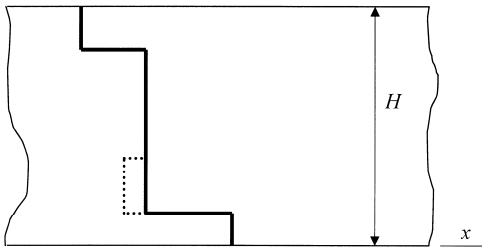
Sastavni deo algoritma čini i stalno ispitivanje da li je neki pravougaonik previše "ištrčao", a ako se to desi, onda se u narednom "redu" ne vrši ređanje pravougaonika uz "segment" koji predstavlja desnu granicu tog "ištrčalog" pravougaonika (v. sl.4). Time se obezbeđuje "ravnomerno" napredovanje "fronta" (veličina dopuštenog "štrčanja" zadaje se, u istim jedinicama u kojima su zadate veličine stranica pravougaonika i ostali parametri vezani za dužinu, pomoću "iskustvenog" parametra *SPIC*).

*)Na ovoj, kao i na ostalim slikama u radu, **punom linijom** će biti označavan prethodni „front" − granica do koje se došlo pri ređanju pravougaonika u prethodnom „redu".



Slika 4.

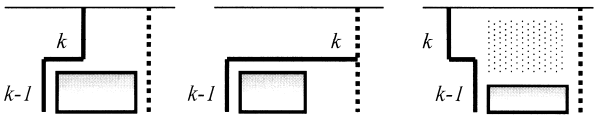
Osim poravnavanja "fronta" (u slučaju zaostalih "džepova") i kontrole "štrčanja", uvedena je i korekcija "fronta" u slučaju kad se susedni, a međusobno paralelni, "segmenti" ne razlikuju *mnogo* (to je opet stvar "eksperimentalnog" određivanja sada veličine *PORA*, kojom će se regulisati ovakvo poravnavanje "fronta"); time što se koriguje "front", tj. smanjuje broj "segmenata" (v. sl.5, gde je nizom simbola ". . . . ." označen deo prvobitnog "fronta" koji je "pomaknut"), trebalo bi da se postigne bolje "pakovanje" uz njega u sledećem "redu".



Slika 5.

Ovim je, u osnovi, završeno opisivanje treće faze heurističkog algoritma za optimizaciju rasporeda crteža. Dalje će biti reči pre svega o nekim detaljima vezanim za pomenuta poravnavanja, ali i za još neke druge korekcije "fronta". Te korektivne procedure (koje sprečavaju da se javi "front" od preteranog broja "segmenata" i od ključnog su značaja za dobijanje "optimalnog" rešenja) biće sada opisane na mogućim slučajevima pri "slaganju" pravougaonika.

Zadržimo se najpre na pitanju šta treba da se uradi kad uz neki "segment" prethodnog "fronta" ne može da se postavi ni jedan od preostalih pravougaonika. Razlikuju se dva takva slučaja: kad je u pitanju poslednji "segment" prethodnog "reda" ili kad je u pitanju neki "međusegment"**).



Slika 6.

Ako se uz poslednji "segment" prethodnog "reda" ne može postaviti ni jedan od preostalih pravougaonika, tada

**) Zbog usvojene *strategije* da se stalno biraju pravougaonici *najvećih mogućih* dimenzija koji mogu stati na "tekućoj" poziciji, tj. uz "tekući" segment, ne može se desiti da uz prvi "segment" prethodnog "reda" ne može da stane ni jedan od preostalih pravougaonika!

mogu da nastupe situacije prikazane na sl.6, pri čemu je nizom simbola " ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ " označen i "nastavak" novog "fronta", tj. desna granica novog "reda". Odgovarajući deo programa glasi:

```
if(k.eq.NSEG) then
        YYT(2,NSEGT)=H
        XXT(NSEGT)=max(XXT(NSEGT),XX(k))
        ...
```

Treba naglasiti da se u trećem slučaju na slici 6*), pre produžavanja "fronta", vrši ispitivanje da li se možda neki od preostalih pravougaonika može staviti u "plići", ali "širi" "džep" (označen simbolima "▒").

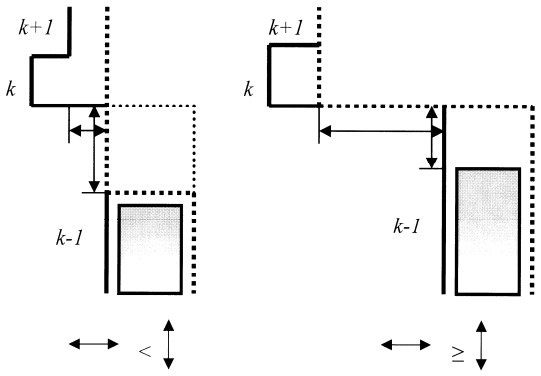
Ako, pak, *k*-ti "segment" (*k>1*), uz koji ne može da se postavi ni jedan od preostalih pravougaonika, nije poslednji "segment" u prethodnom "redu", onda se mogu razlikovati slučajevi prikazani na slikama 7a i 7b**), ***). Deo programa u kom se vrši ta korekcija "fronta" je, za *XX(k)* ≤ *XX(k+1)*, oblika:

```
YY(1,k+1)=YYT(2,NSEGT)
if(XX(k-1).le.XX(k+1)) then
        YYT(2,NSEGT)=YYT_OLD
        YY(1,k+1)=YYT(2,NSEGT)
else
        if((XX(k-1)-XX(k+1)).lt.(YY(2,k-1)-
YYT_OLD)) then
                XX(k+1)=XX(k-1)
                YYT(2,NSEGT)=YYT_OLD
                YY(1,k+1)=YYT(2,NSEGT)
        endif
```

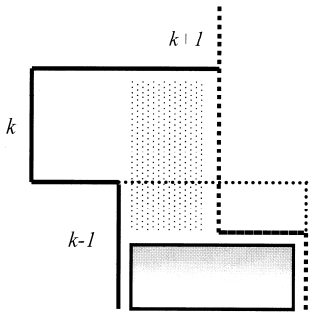
odnosno, za *XX(k) > XX(k+1)*, oblika

```
XX(k+1)=XX(k)
YY(1,k+1)=YYT(2,NSEGT)
if((XX(k-1)-XX(k+1)).lt.(YY(2,k-1)-YYT_OLD).and.
                        XX(k-1).gt.XX(k+1)) then
    XX(k+1)=XX(k-1)
    YYT(2,NSEGT)=YYT_OLD
    YY(1,k+1)=YYT(2,NSEGT)
endif
```

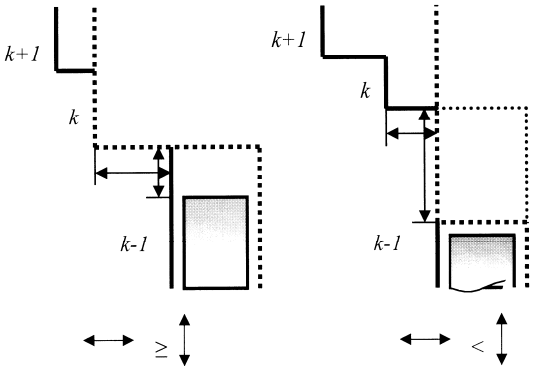
gde je *YYT_OLD (=YYT(2,NSEGT))* zapravo stvarna vrednost (tj. vrednost određena pre eventualne korekcije zbog ažuriranja novog "fronta") *Y*-koordinate gornje tačke *NSEGT*-og "segmenta" u novom "frontu".



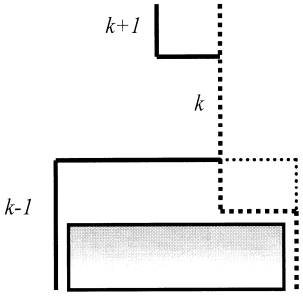
XX(k) ≤ XX(k+1)
XX(k-1) > XX(k+1)



Slika 7a.



XX(k) > XX(k+1)
XX(k-1) > XX(k)



XX(k) > XX(k+1)
XX(k-1) ≤ XX(k)

Slika 7b.

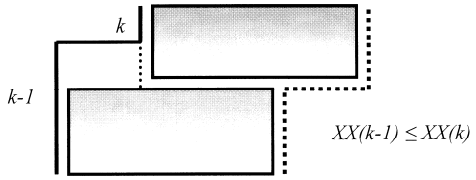
*) Napomenimo da u opisane korekcije "fronta" spada i eventualno produženje "fronta" u prvom "redu" (v. sl.2), koje se vrši pre prelaska na drugi "red".

) Nizom simbola ". . . . .**" na tim slikama je označen deo fronta koji se formira pri "punjenju" uz *k-1*-vi "segment" (v. dalje), ali se to ovde koriguje.

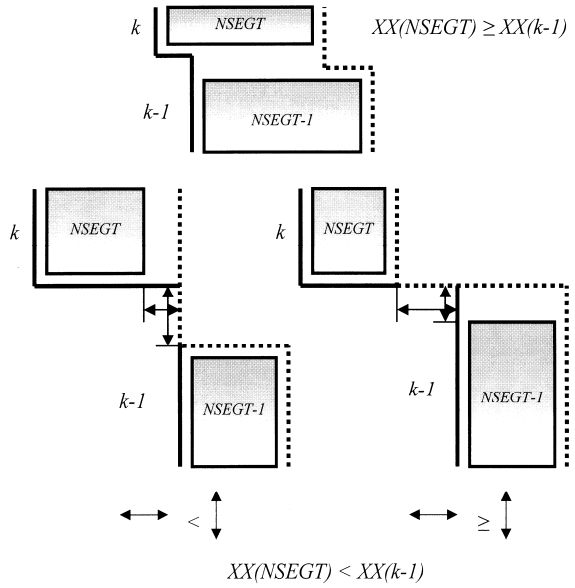
***) Treba naglasiti da se i ovde (v. šta je rečeno uz sl.6), u slučaju kad je *XX(k) < XX(k-1)*, pre "zatvaranja" nekog "džepa", vrši ispitivanje da li se možda neki od preostalih pravougaonika može staviti u "plići", ali (kad je *YYT_OLD < YYT(2,NSEGT))* "širi" "džep" (takav "džep" je "najuočljiviji", zbog *XX(k-1) < XX(k+1)*, na sl.7a, gde je i "popunjen" simbolima "▒").

Do sada je bilo reči o tome šta će da se radi kad na nekom mestu ne može da se "udene" novi crtež; međutim, "peglanje" "fronta" će da se vrši i posle "ubacivanja" novog pravougaonika. Tačnije, vršiće se korekcija dotle određenog novog "fronta". Pošto se koriguje "dotle određeni" front, jasno je da mora biti $k > 1$. Uz to se ispituje i da li je $XX(k) < XX(k-1)$, jer u protivnom (kad je $XX(k) \geq XX(k-1)$), nema potrebe za korekcijom (v. sl.8) onog što je za $k-1$ već urađeno pri pokušaju "punjenja" "džepova" (o tome će dalje biti reči), tj. "front" je već bio korigovan (tačnije, "segment" k je bio "spušten" još u $k-1$-om koraku). Razlikovaćemo slučajeve*) prikazane na sl.9, a što se tiče odgovarajućeg dela programa, on je oblika:

```
if(XX(k).lt.XX(k-1)) then
      if(XXT(NSEGT).lt.XXT(NSEGT-1)) then
            if(XXT(NSEGT).ge.XX(k-1)) then
                  YYT(2,NSEGT-1)=YYT_OLD
                 YYT(1,NSEGT)=YYT(2,NSEGT-1)
            else
                  if((XX(k-1)-XXT(NSEGT)).lt.(YY(2,k-1)-YYT_OLD)) then
                        YYT(2,NSEGT-1)=YYT_OLD
                        YYT(1,NSEGT)=YYT(2,NSEGT-1)
                       XXT(NSEGT)=XX(k-1)
                  endif
            endif
      endif
endif
```



Slika 8.

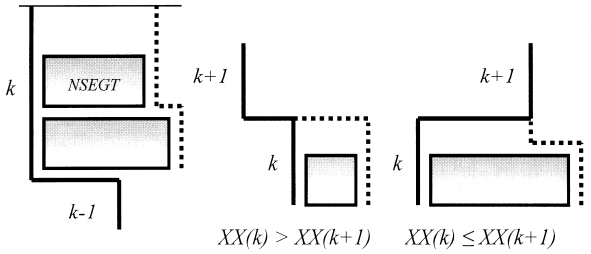


Slika 9.

*) Što se tiče slučaja kad je $XXT(NSEGT) \geq XXT(NSEGT-1)$, nema potrebe posebno ga analizirati, jer se on ne može "poboljšati" u odnosu na već učinjenu korekciju.

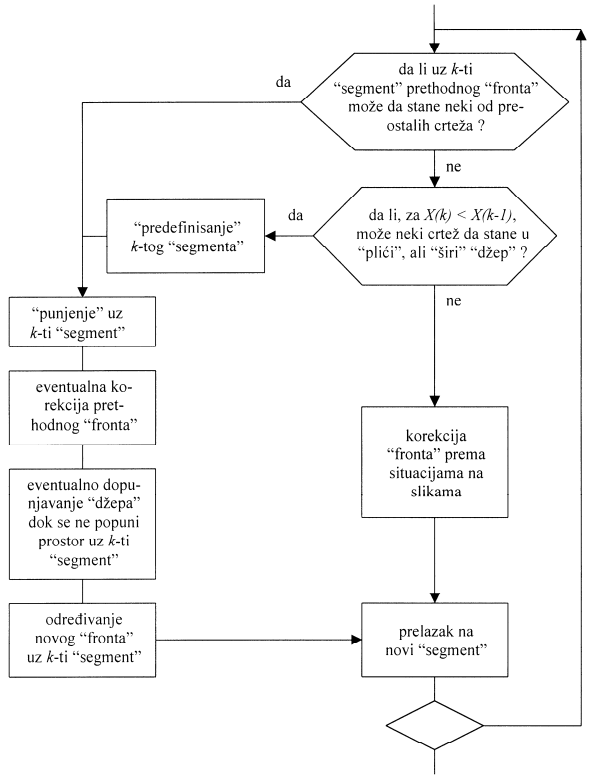
Time su iscrpljene mogućnosti korekcije "fronta" posle "ubacivanja" crteža uz k-ti "segment" prethodnog "fronta".

Međutim, može se desiti da se uz k-ti segment može postaviti ne samo jedan, nego i još jedan (ili više) pravougaonik(a), tj. da se iskoristi postojeći "džep", a kad se to uradi onda treba pristupiti i ažuriranju "fronta" zbog preostalog "džepa". Pri tome treba razlikovati situaciju kad je $k=NSEG$ (kada se jednostavno "produži" "front" do širine trake; v. sl.10a) i kad je $k < NSEG$ (sl.10b), s tim što je sad učinjena korekcija "fronta" privremena, jer kasnije može biti izvršena ponovna korekcija "fronta" (prema onom na sl.9).



Slika 10a. **Slika 10b.**

Time je završeno opisivanje onih slučajeva u kojima je usvojeno da će se vršiti korigovanje "fronta" prilikomređanja pravougaonika u jednom "redu". Sama šema "punjenja" uz tekući (k-ti u "frontu") "segment" prikazana je na sl.11.



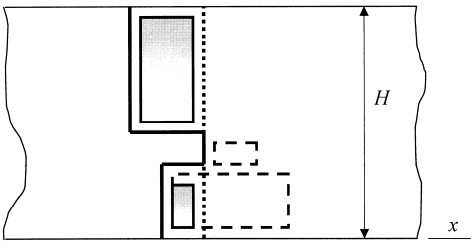
Slika 11. Šema popunjavanja uz tekući „segment"

Korekcija poslednjeg "reda"

Izvesno poboljšanje iskorišćenosti crtaće površine može se, u određenim slučajevima, dobiti kad se posle završenog celog postupka ispita da li je u poslednjem "redu", zbog

njegove nepotpunosti, moguće izvršiti izvesno "premeštanje" pravougaonika*) kako bi se bolje "iskoristili" neki "džepovi" u pretposlednjem "redu" do kojih poslednji "red" nije ni "došao" (v. sl.12, gde su nizom simbola "– – –" označeni položaji poslednja dva pravougaonika posle izvršene korekcije njihovog razmeštaja). Stoga se predlaže sledeći algoritam za eventualno korigovanje poslednjeg "reda" (tj. za smanjivanje "štrčanja" u poslednjem "redu"):

- naći u poslednjem "redu" pravougaonik koji najviše "štrči"
- ispitati da li taj pravougaonik može da stane na neko od slobodnih mesta uz front (uz eventualnu rotaciju), ali tako da se to "štrčanje" smanji
- ako takvog položaja nema, postupak je završen, ali
- čim**) se taj položaj nađe (bez obzira na to da li možda ima i daljih "džepova" gde bi to "udevanje" bilo bolje), pravougaonik se u njega premešta, ažurira se poslednji "red", pa se opet traži koji sad crtež u njemu najviše "štrči" itd.



Slika 12.

Na pojedinačne primere iz kojih se vidi kako je izvršeno to "razmeštanje" pravougaonika u poslednjem "redu" biće ukazano kasnije.

Testiranje heurističkog postupka optimizacije

Već pri prvim testiranjima programa dobijenog implementiranjem opisanog heurističkog postupka moglo se uočiti da on radi *brzo****) (što se moglo i očekivati, budući da se radi o direktnoj, a ne nekoj iterativnoj proceduri). Brz rad programa naveo je na ideju da se oformi i varijanta koja, u beskonačnom ciklusu, generiše random-generatorom ulazne podatke o grupi pravougaonika, raspoređuje ih i potom izračunava procenat neiskorišćenosti crtaće površine. Pre nego što bude reči o detaljima karakterističnim za tu varijantu, napomenimo da je to omogućilo da se ispitaju desetine hiljada slučajeva, uz variranje ulaznih parametara, pa, iako to nema snagu dokaza****), treba reći da se prosek neiskorišćenosti površine za pojedine grupe slučajeva kretao od 9% do 15%; to je znatno manje od prosečnih 40% do 50% neiskorišćenosti skupe uvozne hartije prilikom crtanja na ploteru.

Random-generator ulaznih podataka

Prilikom testi-ranja heurističke optimizacije, ulazni podaci su generisani tako što im je zadavan željeni opseg (najmanji i najveći mogući broj pravougaonika *N_MIN* i *N_MAX*, njihove najmanje i najveće moguće dimenzije *RECT_MIN* i *RECT_MAX*), a potom su, pomoću *random*−generatora*****), u svakom prolazu dobijani i broj crteža *N* i njihove dimenzije, što je programski realizovano na sledeći način:

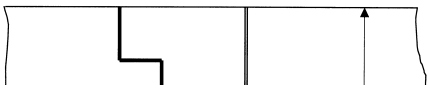
```
integer seed
real mth$random
          ...
          seed=for$secnds(0.)
          ...
          N=mth$random(seed)*(N_MAX-N_MIN)+N_MIN
          ...
do i=1,N
        A=mth$random(seed)*(RECT_MAX-RECT_MIN)+RECT_MIN
        B=mth$random(seed)*(RECT_MAX-RECT_MIN)+RECT_MIN
        ...
enddo
```

Generisani podaci su se zatim koristili za određivanje procenta neiskorišćenosti crtaće površine.

Procentualno određivanje neiskorišćenosti crtaće površine

Procenat neiskorišćenosti crtaće površine dobijamo kad polutraku "presečemo" tangentom na stranicu "najisturenijeg" pravougaonika u poslednjem "redu" (v. sl.13, gde je taj "presek" označen dvostrukom linijom), pa od tako dobijene pravougaone površine oduzmemo površinu svih ("presortiranih") crteža. Ukupna površina svih pravougaonika *AREA_RECT* određuje se u ciklusu:

```
          AREA_RECT=0.
          ...
do i=1,N
          ...
          AREA_RECT=AREA_RECT+(A(i)-FUGA)*(B(i)-FUGA)
          ...
          enddo
```

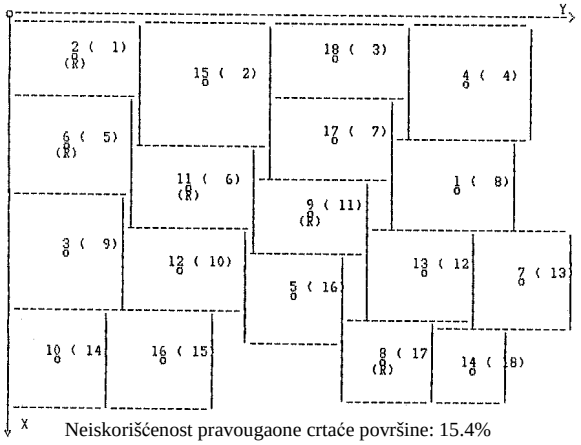


(Pri tome treba primetiti da je ovakav način računanja u suštini "najnepovoljniji", jer se, "poravnavanjem" celog poslednjeg "fronta", u neiskorišćenu površinu ubrajaju i "džepovi" koji se vide na sl.13).

Izračunati procenat neiskorišćenosti crtaće površine za svaku grupu generisanih pravougaonika smeštao se u posebnu datoteku. To je omogućilo da se, po prestanku testiranja rada programa, odredi prosečna neiskorišćenost crtaće površine za sve generisane grupe pravougaonika.

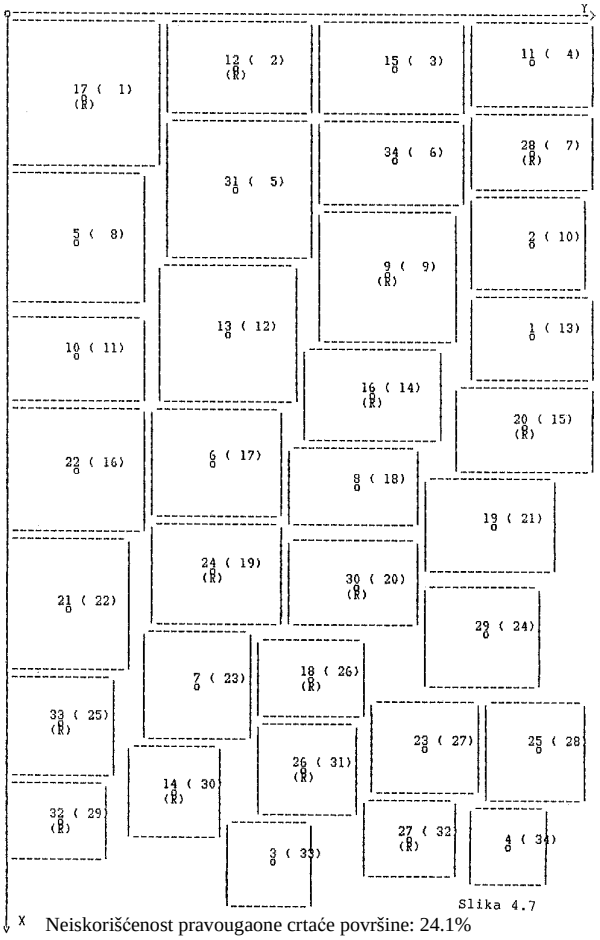
Primeri

Pre nego što budu prokomentarisani neki primeri pseudooptimalnogređanja zadatih pravougaonika u polubeskonačnu traku širine H, treba istaći da je posebno razvijen program za prikazivanje rezultata pseudooptimizacije na alfanumeričkom terminalu. Suština je u tome da se homotetičnim postupkom, u razmeri određenoj odnosom parametra *LPWIDTH* (koji predstavlja širinu trake na ekranu u karakterima) i stvarne širine trake H na ploteru, odrede "ekranske" koordinate temena (a i središta) pravougaonika. "Popunjavanjem" ekrana između tih tačaka



Neiskorišćenost pravougaone crtaće površine: 15.4%

Slika 16.



Neiskorišćenost pravougaone crtaće površine: 24.1%

Slika 17.

Tabela 2

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
a_i	17.	18.	17.	16.	26.	21.	21.	16.	28.	17.	16.	29.	27.	19.	18.	28.	30.	21.
b_i	25.	23.	17.	16.	28.	26.	22.	26.	26.	28.	24.	18.	28.	18.	29.	18.	29.	16.

19	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*
19.	28.	25.	25.	19.	26.	20.	19.	19.	24.	21.	26.	27.	20.	22.	16.
25.	16.	25.	27.	21.	20.	20.	17.	15.	15.	22.	18.	29.	15.	20.	29.

Završne napomene, zaključci i buduće aktivnosti

U radu je iznet heuristički postupak pseudooptimalnog razmeštanja proizvoljnog broja pravougaonika u polubeskonačnoj traci zadate širine. Postupak se, u osnovi, sastoji uređanju pravougaonika u pravcu širine trake, ali tako da se u raspoloživi deo širine trake postavlja onaj od preostalih pravougaonika s najvećom stranicom koja u taj deo može da stane. Međutim, razvijene su i određene korektivne procedure koje sprečavaju da se javi "front" od preteranog broja "segmenata"; to "poravnavanje fronta", koje je i ključno za dobijanje "optimalnog" rešenja, opisano je na mogućim slučajevima pri "slaganju" pravougaonika. Treba naglasiti da se, po potrebi, priređanju pravougaonika vrši i njihova rotacija za $\pi/2$. Postoji i mogućnost zadavanja najmanjeg rastojanja ("fuge") između pravougaonika.

Osnovna odlika programa dobijenog implementiranjem takvog heurističkog postupka jeste (što je i opšta odlika heurističkih programa) da daje *zadovoljavajuće rezultate*, i to *veoma brzo* (jer se radi o direktnoj, a ne nekoj iterativnoj proceduri). Brz rad programa naveo je na ideju da se oformi i varijanta koja, u beskonačnom ciklusu, generiše *random*- -generatorom ulazne podatke (u željenom opsegu: najmanji i najveći broj pravougaonika, njihove najmanje i najveće dimenzije) o grupi pravougaonika u jednom "prolazu", raspore

[18] GOULIMIS,C. Optimal solutions for the cutting stock problem. European Journal of Operational Research, 1990, vol.44, no.2, pp.197-208.

[19] DOWSLAND,K.A. Efficient automated pallet loading. European Journal of Operational Research, 1990, vol.44, no.2, pp.232-238.

[20] FARLEY,A.A. Selection of stockplate characteristics and cutting style for two dimensional cutting stock situations. European Journal of Operational Research, 1990, vol.44, no.2, pp.239-246.

[21] FARLEY,A.A. The cutting stock problem in the canvas industry. European Journal of Operational Research, 1990, vol.44, no.2, pp.247-255.

[22] OLIVEIRA,J.F., FERREIRA,J.S. An improved version of Wang's algorithm for two-dimensional cutting problems. European Journal of Operational Research, 1990, vol.44, no.2, pp.256-266.

[23] BISCHOFF,E.E., MARRIOTT,M.D.. A comparative evaluation of heuristics for container loading. European Journal of Operational Research, 1990, vol.44, no.2, pp.267-276.

[24] GEHRING,H., MENSCHNER,K., MEYER,M. A computer-based heuristic for packing pooled shipment containers. European Journal of Operational Research, 1990, vol.44, no.2, pp.277-288.

Rad primljen: 9.5.2002.god.

Heuristic optimization of rectangle nesting in the semi-infinite band of the given width

A heuristic procedure for the pseudooptimal rectangle nesting in the semi-infinite band of the given width is proposed. The basic feature of the corresponding program is to give satisfactory results and to give them very quickly.

Key words: **optimization, heuristic optimization, allocation, nesting,plotter, semi-infinite band, rectangle.**

Optimisation heuristique du 'nesting' des rectangles sur un ruban sémi-infini de largeur donné

Un procédé heuristique du 'nesting' pseudo-optimal des rectangles sur un ruban sémi-infini de largeur donnée est proposé. La caractéristique principale du programme obtenu par l'implementation de ce procédé est de donner les résultats satisfaisants et, qui plus est, de les donner très vite.

Mots-clés: **optimisation, optimisation heuristique, allocation, 'nesting', traceur, ruban sémi-infini, rectangle.**